CONNECT

Emergent Connectors for

Eternal Software Intensive Networked Systems

ICT FET IP Project

# Deliverable D6.4

# **Assessment Report**

*Experimenting with CONNECT in Systems of Systems, and Mobile Environments*

SEVENTH FRAMEWORK PROGRAMME

http://www.connect-forever.eu

Inria · docomo DOCOMO Euro-Labs · mblentic · LANCASTER UNIVERSITY · THALES · Universita' dell'Aquila · technische universität dortmund · University of Oxford · Uppsala Universitet

| | | |
|---|---|---|
| **Project Number** | : | 231167 |
| **Project Title** | : | CONNECT – Emergent Connectors for Eternal Software Intensive Networked Systems |
| **Deliverable Type** | : | Report |

| | | |
|---|---|---|
| **Deliverable Number** | : | D6.4 |
| **Title of Deliverable** | : | Assessment report |
| **Nature of Deliverable** | : | R/P |
| **Dissemination level** | : | PU |
| **Internal Document Number** | : | V0.3 |
| **Contractual Delivery Date** | : | 30 November 2012 |
| **Actual Delivery Date** | : | 17 December 2012 |
| **Contributing WPs** | : | WP6 |
| **Editor(s)** | : | Youssouf Mhoma (TCF) |
| **Author(s)** | : | Emil Andriescu (Ambientic-Inria), Amel Bennaceur (Inria), Antonia Bertolino (CNR), Antonello Calabro (CNR), Paul Grace (LANCS), Malte Isberner (TUDO), Antoine Léger (TCF), Maik Merten (TUDO), Youssouf Mhoma (TCF), Pierre Châtel (TCF), Charles Morisset (CNR), Animesh Pathak (Inria), Pierre-Guillaume Raverdy (Ambientic), Rachid Saadi (Ambientic), Roberto Speicys-Cardoso (Ambientic), Daniel Sykes (Inria). |
| **Reviewer(s)** | : | Gordon Blair (LANCS), Valérie Issarny (Inria) |


# Abstract

The core objective of WP6 is to evaluate the CONNECT technologies under realistic situations. To achieve this goal, WP6 concentrated a significant amount of its 4th year effort on the finalization of the implementation of the GMES scenario defined during the 3rd year. The GMES scenario allows the consortium to assess the validity of CONNECT claims and to investigate the exploitation of CONNECT technologies to deal with the integration of real systems. In particular, GMES requires the connection of highly heterogeneous and independently built systems provided by the industry partners. WP6 contributed also in providing mobile collaborative applications and case studies showing the exploitation of CONNECTORs on mobile devices.



# Keyword list

Assessment, CONNECTor, CONNECTability, Experiment, GMES, CONNECT architecture, CONNECTor algebra, Mediator synthesis, Protocol learning.

# Document History

| Version | Type of change | Author(s) |
|---|---|---|
| 0.1 | Document outline | Youssouf Mhoma (TCF)<br>Rachid saadi (Ambientic)<br>Pierre-Guillaume Raverdy (Ambientic) |
| 0.2 | Initial contributions | All |
| 0.3 | Revised contributions | All |
| 1.0 | Overall edition | Youssouf Mhoma (TCF) |
| 2.0 | Revision based on reviews | All |
| 3.0 | Final version | Youssouf Mhoma (TCF) |

# Document Review

| Date | Version | Reviewer | Comment |
|---|---|---|---|
| 30 Nov. 2012 | 1.0 | Gordon Blair | More detail on experiments needed |
| 11 Dec. 2012 | 2.0 | Valerie Issarny | Editorial comments |

# Table of Contents

# 1 Introduction

The scope of WP6, as introduced in the Description of Work (DoW), is: "*to assess the CONNECT architecture and prototypes, as generated by WP1, against actual scenarios*". To this extent, the work performed within WP6 has concentrated on:

- Elaborating industry-strength scenarios, further identifying real systems that are available to the project members and that can provide a testbed against which to experiment with CONNECT technologies.
- Implementing use cases pertaining to the scenarios, in particular exploiting CONNECT enablers to make interoperable the legacy networked systems involved in the use cases. Part of the work also includes implementing a number of networked systems.

While the elaboration of scenarios was mostly carried out in the previous reporting periods, the 4th year work was focused on use cases implementation, further enabling to assess the overall CONNECT architecture and related enablers. Although the assessment work is integral part of WP6 achievements, it is actually reported directly next to the scientific and technology content of deliverables D1 to D5, as a way to be more convenient and ease traceability for the readers.

This report then concentrates on the description of the use cases that were implemented and that pave the way for exploitation of CONNECT technologies by the industry partners. Use cases specifically relate to the following areas:

- **Systems of Systems**, where we more specifically focus on using CONNECT for addressing the interoperability requirements raised by scenarios related to the European programme on *Global Monitoring for Environment and Security.*
- **Mobile collaborative applications***,* where we focus on exploiting CONNECT to address the interoperability requirements that arise for mobile applications deployed on increasingly heterogeneous mobile platforms and interacting with diverse Internet-based services, among which Cloud services. Then, we addressed both application-to-application and application-to-Internet services scenarios.

## 1.1 Summary of Y4 achievements

In a nutshell, WP6 achievements over Year 4 relate to:

- Delivery of the final evaluation platform.
- Finalizing the implementation of the Joint Forest-Fire Operation use case (simply referred to as firefighting use case) that is part of the GMES scenarios, which includes the implementation of all the networked systems involved in the use case.
- Implementation of new use cases in the area of mobile collaborative applications.
- Application of the assessment methodology introduced in Deliverable D6.3, whose outcome is reported in Deliverables D1 to D6 according to the specific focus of the assessment.

## 1.2 Third review recommendations

The project's 3rd review provided one main recommendation relevant to the work to be performed in the 4th year within WP6:

- *"Demonstrators need to be extended in order to show the full potential and extent of the various CONNECT enablers. In doing so, the need for fully distributed enablers and the potential effects of communication latencies should be clarified."*

The recommendation was accounted for in the further development of the GMES use case as well as in the development of the mobile collaborative applications exploiting the mobile

version of CONNECT. Related updated Networked Systems are formally defined in the Appendix.

## 1.3 Challenges for Year 4

The main challenges for Year 4 work on the GMES use case related to:

(i) Effectively integrating the enablers according to the CONNECT architecture so as to support the use case.
(ii) Dealing with non-functional properties in the implementation of the use case.

A number of challenges were also to be faced for supporting the mobile collaborative applications, which related to adapting the CONNECT architecture to the specifics of the mobile environment, as detailed in Deliverable D1.4.

## 1.4 Achievements in Year 4

As suggested above, one of the main achievements of the work package during Year 4 has been to experiment with CONNECT solutions using the firefighting GMES use case. This significantly assisted the consortium in dealing and experimenting with the actual integration of the enablers, as specified by the CONNECT architecture. In particular, this work is certainly the one that allowed the consortium to produce an effectively working CONNECT prototype. WP6 work further allowed the consortium to thoroughly assess the CONNECT architecture and enablers according to the plan set in Deliverable D6.3. Last but not least, WP6 work paves the way for exploitation of the CONNECT results by industry partners, based on the advanced interoperable services that have been implemented, especially in the mobile domain.

## 1.5 Structure of the deliverable

This report is decomposed in two core parts:

- Section 2 is concerned with the implementation of the firefighting use case using CONNECT to overcome heterogeneity issues.
- Section 3 concentrates on leveraging the CONNECT architecture towards sustaining mobile collaborative applications.

Each section is structured similarly, providing a description of the use cases and then an assessment of the use cases in the specific target exploitation domains, i.e., Systems of systems for GMES, and mobile collaborative applications.

Finally section 4 concludes with a summary of WP6 contributions and resulting exploitation perspectives for CONNECT technologies.

# 2 GMES Use Case

The GMES use case was extensively detailed in Deliverable in D6.3 and we refer the interested reader to this deliverable for a detailed description of the NSs that were implemented and related reliance on CONNECT enablers for their CONNECTion. Still, we would like to stress that the implementation of a number of networked systems had to be finalized during the fourth year, and this is reflected in the Appendix.

Key focus of our work during the reporting period has then been to actually experiment with the CONNECTion of GMES NSs to assess the CONNECT architecture and enablers (see D1 to D5) as well as assess the relevance of the CONNECT approach for the System of Systems domain.

The next section briefly recalls the main features of the GMES use case, then Section 2.2 and finally Section 2.3 outline the resulting assessment of CONNECT to sustain interoperability in the system of systems area.

## 2.1 Use case overview

GMES (Global Monitoring for Environment and Security) is the European Program for the establishment of a European capacity for Earth Observation started in 1998. The services provided by GMES address six main thematic areas: land monitoring, marine environment monitoring, atmosphere monitoring, emergency management, security and climate change.

The emergency management service directs efforts towards a wide range of emergency situations; in particular it covers different catastrophic circumstances: floods, forest fires, landslides, earthquakes and volcanic eruptions and humanitarian crises.

**Within CONNECT, we concentrated on the Forest fire situation.** The scenario illustrates the management of forest-fire, close to a border village and a factory, between country A and country B. Forest monitoring and forest fire management in the country A are the responsibility of the Country A Command and Control fire operations center (C2-A). For example in France, the CODIS, led by a professional fire-fighter, is the regional authority in charge of coordination of operational forces during fires and disasters for one French department. Then, the C2-A center must interoperate with a number of Networked Systems (NSs) of Country B so as to monitor the area (see Deliverable D6.3 for the specification of the NSs, and Appendix for updates on these specifications after Year 4):

- **NS 1** - UAV
- **NS 2** - UGV
- **NS 3.1** - Camera Fixed
- **NS 3.2** - Camera Mobile Main
- **NS 4** - C2 GIS
- **NS 5** - Mobile Weather Station
- **NS 6** - Weather Service
- **NS 7.1** - Positioning System – Country A
- **NS 7.2** - Positioning System – Country B

Figure 2.1 outlines the middleware-layer protocols of the various NSs while Figure 2.2 depicts the CONNECTion we have been focusing on.

| Middleware-layer protocols | | |
|---|---|---|
| Network Systems | Protocol | Exchange Pattern |
| NS 1 UAV | SOAP | RPC |
| | RTSP | Stream |
| NS 2 UGV | SOAP | RPC |
| | MJPEG | Stream |
| NS 3.1 Camera Fixed | DPWS | RPC |
| | RTSP | Stream |
| NS 3.2 Camera Mobile main | DPWS | RPC |
| | MJPEG | Stream |
| NS 4 C2 GIS | SOAP | RPC |
| | MJPEG | Stream |
| NS 5 Weather Station | SOAP | RPC |
| NS 6 Weather Service | SOAP | RPC |
| NS 7.1 Position System A | SOAP | RPC |
| NS 7.2 Position System B | AMQP | Pub/Sub |

**Figure 2.1: NSs and protocols**

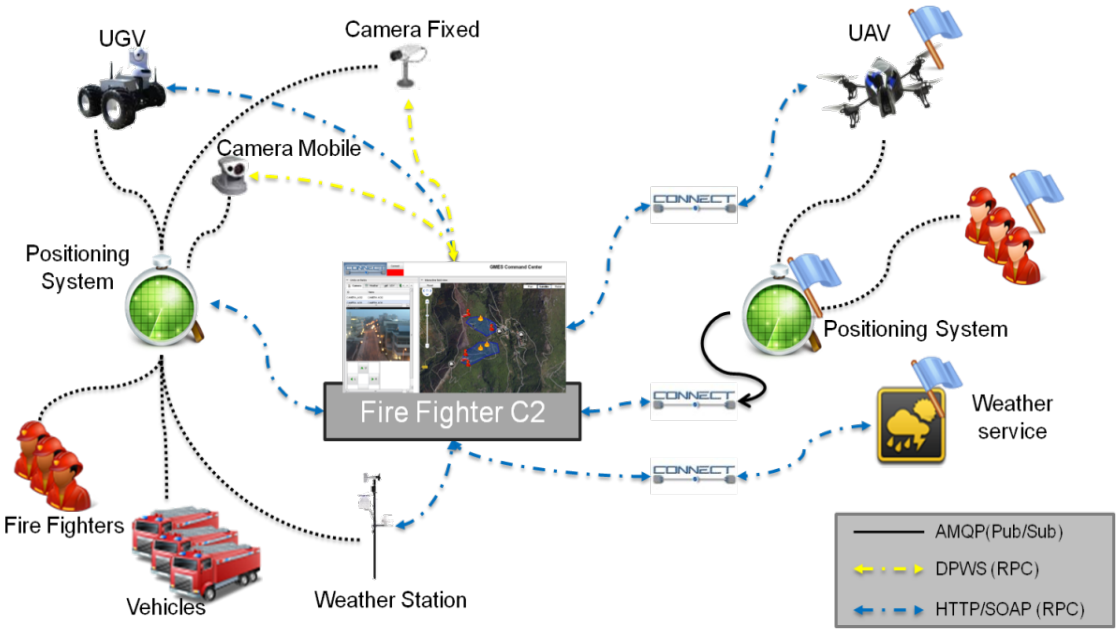


**Figure 2.2:** Use case - NSs & CONNECTions

## 2.2 Focus on Enablers

In the GMES use case, 8 enablers have been integrated and each plays a role at some point of the scenario.

- **E1** - Discovery
- **E2** - Learning
- **E3** - Synthesis
- **E4** - Deployment
- **E5** - Monitoring
- **E6** - Interaction
- **E7** - DePer
- **E8** - Security

Figure 2.3 outlines the sequence of actions chained when a CONNECTor has to be synthesized and deployed between two network systems.

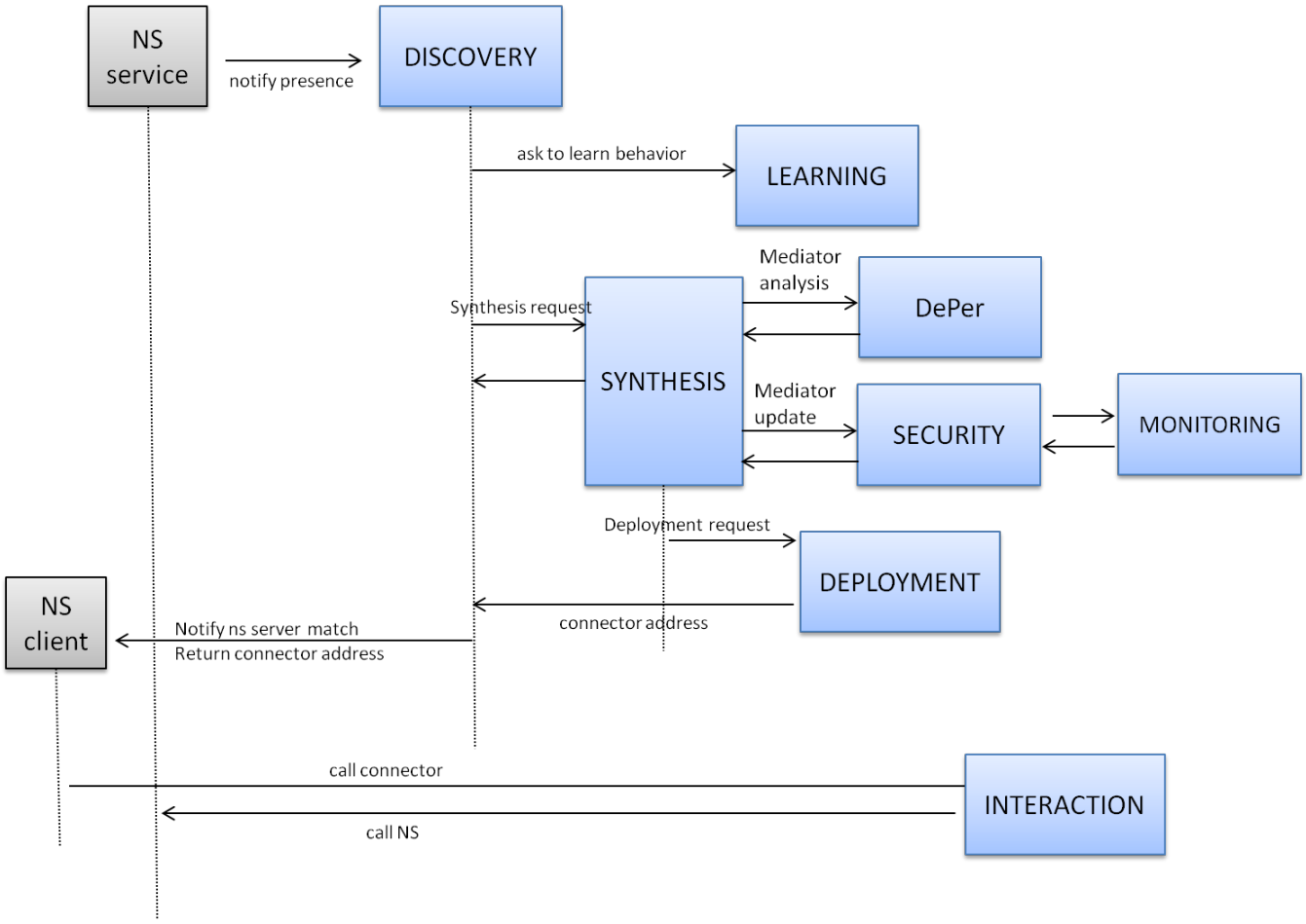


**Figure 2.3:** The interactions between enablers

In the following sections, we describe the realization of the different phases of the CONNECTion process in the context of GMES. Section 2.2.1 outlines what happens during the initial discovery phase, the essential step for handling discovery protocol heterogeneity. Section 2.2.2 outlines the learning supportive phase that complete discovery with non-provided behaviour models. Section 2.2.3 illustrates the synthesis and deployment phases of the GMES CONNECTors. Section 2.2.4 goes deeper into the CONNECTability phase which is triggered by the Synthesis enabler prior to CONNECTor deployment.

### 2.2.1 Discovery Phase

For the GMES scenario, the CONNECT discovery techniques were extensively used in all the networked systems that required interoperability. Specifically, the Fire Fighter C2 system, the UAV service, the positioning system, and the weather service all announced their affordances and interfaces, and optionally, their behavior and non-functional properties, using the CONNECT discovery protocol. Upon receiving these announcements, the CONNECT Discovery enabler invokes the Learning enabler for obtaining the behaviors of the networked systems that did not provide their behavior a priori, and subsequently, invokes the Synthesis enabler for the synthesis and deployment of the CONNECTors where matching required/provided affordances were found (specifically, C2-UAV, C2-positioning system, and C2-weather service). Upon deployment, the CONNECT Discovery enabler responds to the "requiring" networked system (in this case, C2) with the URI of the deployed CONNECTors, in response to the initial discovery request. E.g., when C2 requests for the UGV, the Discovery enabler replies to it with address of the deployed C2-UAV CONNECTor.

### 2.2.2 Learning Phase

In the GMES scenario, developed by WP6 with the goal of showing the interplay of all the developed technologies, learning technology was successfully used to generate behavioral models of components such as a service providing weather data and a service that provides access to a flying drone. In each case, the system could be accurately described as a finite state machine that models the complete input/output-behavior. These behavioral models can be obtained without much delay, including the time inherent to networked system invocation. The accuracy of the learned models was verified by manual inspection. In summary, the learning technology proved its ability to generate accurate models that are adequate for CONNECTor synthesis in a time frame that is suitable for ad-hoc CONNECT scenarios.

### 2.2.3 CONNECTor Synthesis/Deployment Phase

The CONNECT Synthesis Enabler along with Deployment enabler perform the central role of creating the software CONNECTors for each of the heterogeneous cases in the GMES scenario. Specifically, the Synthesis Enabler:

(i) Receives the networked system models for each of the matched endpoint systems,

(ii) Computes the required mapping between their interfaces by reasoning about the semantics of their operations and data,

(iii) Uses the computed mapping, to generate the mediator, in the eLTS format, which ensures their behavioural matching, i.e., that they will interact without errors (e.g., deadlock),

(iv) Translates the eLTS model of the mediator to the concrete deployment format (*k*-coloured automata).

(v) If the Networked System Models includes non-functional requirements, the Synthesis Enabler first passes the generated *k*-coloured automata to the DePeR Enabler and the Security Enabler for analysis and further instrumentation. Otherwise, it directly sens the produced coloured *k*-coloured automata to the Deployment enabler.

Figure 2.4 succinctly describes the mismatches that had to be addressed in each use case of the GMES scenario. In the case of mediating the interaction of the C2 with the Weather Service, the mediator has to map the getWeather operation to the *getHumidity* and *getTemperature* operations, which also involves translating the output data received (Humidity and Temperature) into the input data expected (Weather). In the case of the C2 communicating with Positioning-B, there is a unique operation *getPosition*, and hence there is only a one-to-one mapping, while the Deployment Enabler handles middleware heterogeneity. In the case of interaction between C2 and the UAV, the Synthesis Enabler computes only one-

to-one mappings between the operations required by the C2 (*MoveRight*, *MoveLeft*, *MoveFoward*, *MoveBackward*, *TurnLeft*, and *TurnRight*) and those provided by the UAV, but when generating the mediator, the Synthesis Enabler has to call extra operations (*takeoff* and *land*) to allow the UAV to continue its execution and reach its final state. This extra state has to be taken care of in the produced *k*-coloured automata, which requires extra transition in the bridging automata.

| (vi) Use Case | Application-layer heterogeneity | Middleware-layer heterogeneity |
|---|---|---|
| **C2 - Weather Service** | One-to-many operation and data mapping | None – both use SOAP |
| **C2 - Positioning-B** | One-to-one operation mapping | RPC-Pub/Sub: SOAP client with AMQP publisher |
| **C2 - UAV** | Extra provided operations | None – both use SOAP |

**Figure 2.4:** Type of heterogeneity addressed in each GMES use case

The Deployment Enabler upon reception of the *k*-coloured automata of the mediators, eventually instruments and deploys them in the networking environment and the networked systems successfully complete their tasks, illustrating that the interoperability problems are effectively resolved by CONNECT. In the three cases:

i) For the Fire Fighter C2 system interoperating with the weather service, there were behavioural and data mismatches between the two systems that were handled by the synthesis mapping phase;
ii) For the C2 system with the UAV service, there were behavioural differences between the two systems that were overcome by the synthesis mapping;
iii) For the C2 system with the positioning service, there were differences in the middleware protocols employed (i.e. a publish-subscribe protocol AMQP and an RPC protocol SOAP) that were addressed by the concrete binding procedure provided by the Deployment enabler.

### 2.2.4 CONNECTability Phase

In the GMES use case, the CONNECT architecture is equipped with several enablers in charge to enforce, monitor and enhance the behaviour of the CONNECTor offline and at runtime (see Figure 2.5).

The DePer Enabler is in charge of assisting the synthesis and deployment of a CONNECTor suitable to satisfy non-functional requirements, namely dependability and performance related properties. Its activity is carried on both at pre-deployment (where the synthesized CONNECTor is analyzed and possibly enhanced to meet the stated non-functional requirement before deployment) and at run-time (to refine and adapt the analysis, to cope with uncertainties and inaccurate knowledge available at pre-deployment time, as well as evolution undertaken by the networked systems and the environment). The run-time analysis is performed by synergically exploiting cooperation with the Monitoring Enabler, through which operational data of interest to the DePer analysis are gathered and examined. The GMES use case was the ground to provide an example of DePer activities and its integration/cooperation with other Enablers, especially the Monitoring Enabler. In the first client to service interaction (C2-UAV), the two analyzed indicators are: i) a measure of latency, determined as the time from when the C2 client sends one of the possible orders to move (orderToMove) to when it receives an acknowledgement back (orderToMoveACK), and ii) a measure of coverage defined as the percentage of stream video the C2 client correctly receives from the UAV, with respect to the number of video requests made. In the second client to service interaction (C2-Weather Service), the indicator of performance is the latency requirement defined by the C2 client to receive an acknowledgement from the Weather Service once a request is made, while the

coverage requirement defined by the C2 client consists in correctly receiving at least 90% of the requested weather data.

The Security Enabler is in charge of checking that the security policies are respected at runtime. In particular, the CONNECTor between the C2 client and the UAV has to ensure that the UAV does not go into a forbidden area (thus simulating some no-fly zone). Each command from the C2 that would violate the policy is effectively ignored. In order to get the precise location of the UAV, the Security Enabler establishes a separate connection with the UAV and interacts frequently with the CONNECTor to update the position.

The Monitoring Enabler, once started, waits for input to start monitoring non-functional properties provided by one of the two Enablers involved in CONNECTability: DePer Enabler or Security Enabler. At Synthesis-time, probes able to send messages coming through the CONNECTor are inserted. When the DePer Enabler finishes the off-line analysis of the CONNECTor, it sends to the Monitoring Enabler a set of NF-PROPERTIES that have to be respected at runtime: i) latency between two actions: "login – takeoff" on the UAV client has to be < than 10s ii) the maximum amount of messages sent in a window time of 30 seconds can not be more than 500. In the GMES scenario, when the Monitoring Enabler notices that the warning threshold is reached, it notifies all the others Enablers to try reducing the amount of messages sent to the CONNECTor in order to deal with the possible performance lack. The GMES use case illustrates this cooperative network behaviour by making the Security Enabler managing the Monitoring notification: the Security Enabler checks every 10 milliseconds the position of the UAV that is stored inside the CONNECTor; when the Security Enabler receives the "invitation" to reduce the amount of messages, it increases the latency between the polling on the variable stored into the CONNECTor in order to reduce the global amount of messages/traffic of the CONNECTor.

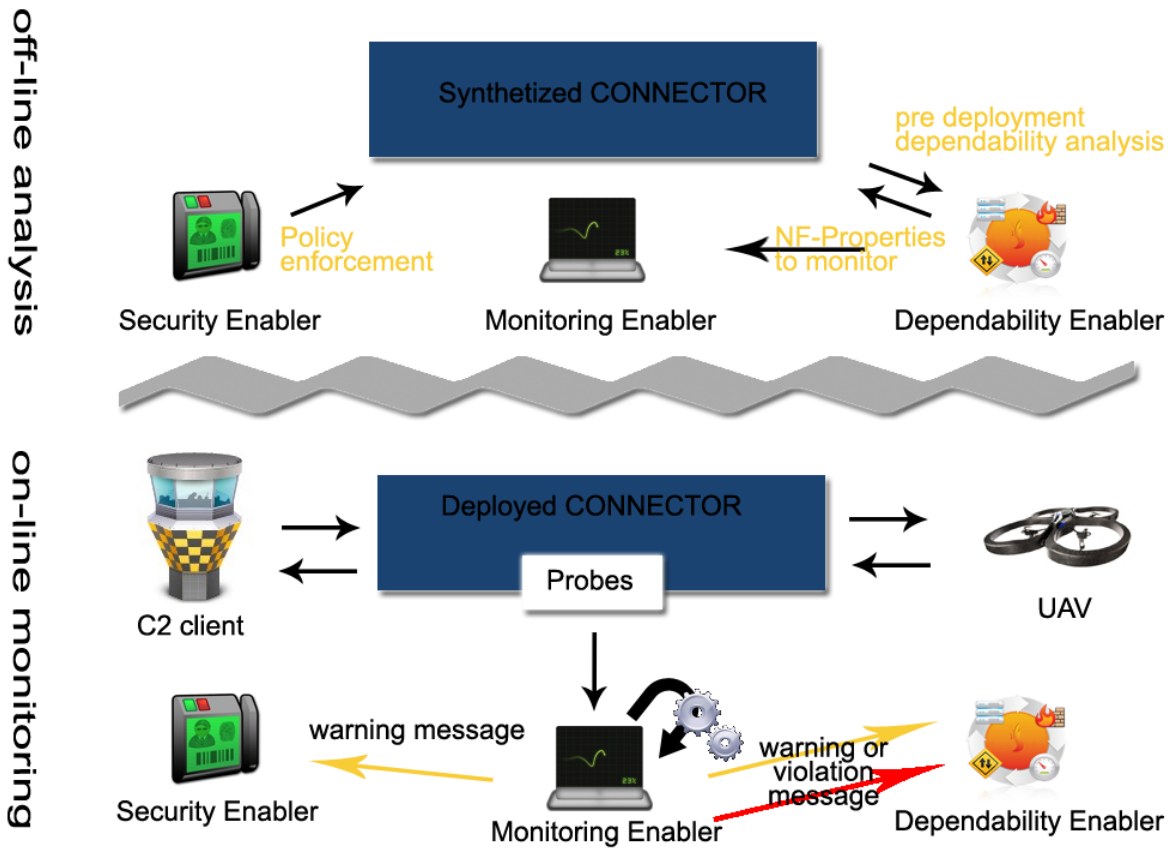


**Figure 2.5:** Dependability, Security and Monitoring cooperation

## 2.3 System of Systems assessment

Based on the experiment using the GMES use case, we have been assessing the possible exploitation of CONNECT for the system of systems domain.

As defined in Wikipedia's page on Systems of Systems[1], several traits are inherent to System of Systems:

- Operational Independence of Elements;
- Managerial Independence of Elements;
- Evolutionary Development;
- Emergent Behaviour;
- Geographical Distribution of Elements;
- Inter-disciplinary Study;
- Heterogeneity of Systems; and
- Networks of Systems.

While some of them are not relevant to CONNECT (such as Inter-disciplinary Study), the others have been gathered in four main objectives given the commonalities of their assessment methodology:

1. **Operational and Managerial Independence of Systems;**
2. **Evolutionary Development and Emergent Behaviour;**
3. **Heterogeneity of Systems; and**
4. **Networks of Systems and Geographical Distribution of Elements.**

The following sections assess CONNECT against each of the above objectives in turn.

### 2.3.1 Respecting the operational and managerial independence of systems

| | |
|---|---|
| **Criteria** | The non-intrusiveness of the CONNECT platform over the networked systems. |
| **Initial assessment** | CONNECT artifacts need to be as transparent as possible to the business networked entities so as not to violate the separation of concerns (SoC) between technical (transport) and business aspects of Systems of Systems (SoS). Confirm independency in the GMES use case. |
| **Contribution** | CONNECT artifacts behave as transparently as possible, in a proxy manner. |
| **Future work** | Even more decoupling between business and technical levels of SoS, still relying on CONNECT. |

*Methodology:*

Within the GMES scenario, the evaluation consists of checking that a networked system implementation's independent existence is not put into question when deployed in a CONNECT context.

*Assessment*:

In the GMES use case, at some point, Country B is responsible of and uses weatherService-B that provides meteo data. Its implementation has no link with CONNECT and does not possess any code that is related to it. When Country B wants to provide its service to Country A, the discovery of the service by CONNECT is managed outside it, without changing the service implementation. The resulting created relation of C2-A←→CONNECTor←→weatherService-B

[1] http://en.wikipedia.org/wiki/System_of_systems

exists, allowing Country A to use weatherService-B. But at the same time, weatherService-B is always available to Country B that keeps the control and the responsibility of weatherService-B.

Thus, we confirm that networked systems can evolve and be used without depending on CONNECT platform, while at the same time being used through CONNECT.

### 2.3.2 Allowing the evolutionary development and emergent behaviour

| | |
|---|---|
| **Criteria** | Support of Evolutionary development. |
| **Initial assessment** | SoS typically does not support dynamic adaptation and evolution due to structural and architectural constraints. The addition and removal of networked systems need to be directly supported by the Discovery Enabler. |
| **Contribution** | The CONNECT platform supports runtime evolution and context change –including adding and removing Networked Systems on the fly. |
| **Future work** | Even more dynamicity in CONNECTors generation at runtime. |

*Methodology:*

Within the GMES scenario, the evaluation consists of:

- Adding and removing Networked Systems to and from the System of Systems, taking specific care of adding and removing Systems that involve different CONNECTors.
- Evolving the code of the C2 system in order to command in a different way the Networked Systems, hence establishing a new, emergent, behaviour.

*Assessment*:

The addition and removal of networked systems is directly supported by the Discovery Enabler.

We further managed to demonstrate the evolutionary development. When a new version of a Network System is developed, the previous version can be removed from CONNECT and the new version added without any consequences thanks to the use of stateless services. For example, in the C2 graphical application, when the weather service is discovered, it appears in the list of available services. When the weather service is shutdown, it is removed from the available services list. Then a new version can be discovered and used again.

### 2.3.3 Supporting systems heterogeneity

| | |
|---|---|
| **Criteria** | To deal with systems heterogeneity through CONNECT. |
| **Initial assessment** | Systems heterogeneity is assessed through the number of supported exchange patterns and the number of transport protocols supported. |
| **Contribution** | Rely on the defined set of exchange patterns and associated transport protocols to support systems heterogeneity in the GMES context. |
| **Future work** | Even more systems heterogeneity support in CONNECT framework. |

*Methodology:*

Within the GMES scenario, a set of exchange patterns and associated transport protocols have already been selected. The evaluation thus consists of checking their effective realization and integration when executing the scenario.

Ambientic's video stream adaptation is also a key contributor to this objective.

*Assessment*:

With the GMES use case, we experienced the support of services using three different kinds of transport protocols and two different exchange patterns. Coping with the bridging between these services is the responsibility of CONNECTors. Following is a reminder of bridging occurring in GMES:

| Client and affordance | Transport Protocol/Exchange Pattern | NS to connect |
|---|---|---|
| **C2 (VehicleWithVideo)** | SOAP-HTTP/RPC | NS1 UAV and NS 6 Weather Service (SOAP-HTTP/RPC) |
| **C2 (VideoSource)** | SOAP-HTTP/RPC | NS3.x Camera (DPWS/RPC) |
| **C2 (PositioningSource)** | SOAP-HTTP/RPC | NS8 Position System B (AMQP/Pub-Sub) |

Ambientic's contribution allows also to transform video stream format and so enrich CONNECT with transformation capabilities displayed in the next charts.

| Video client | Format | Video service |
|---|---|---|
| **C2** | MJPEG | NS1 UAV (RTSP) |
| **Mobile** | RTSP | Camera (MJPEG) |

### 2.3.4 Networks of systems and geographical distribution of elements

| | |
|---|---|
| **Criteria** | To deal with geographical distribution of elements in networks of systems. |
| **Initial assessment** | Geographical distribution is assessed through the density of the networks of supported systems, as well as the effective distribution of the networked systems in a CONNECT architecture. |
| **Contribution** | Implement SoS flexibility and elasticity based on geographical distribution. |
| **Future work** | Put forward geographical distribution of elements as a key "non-functional" property when integrating them in System of Systems. |

*Methodology:*

The evaluation of the first criterion consists of running various alternative ways to start the networked systems (gradually, by set, all at the same time) with a large number of CONNECTed mock-ups of networked systems and see how the System of Systems react.

Regarding the second criterion, the weather service of the demonstration is to be available on the Internet and networked systems are to be deployed at different locations.

*Assessment*:

**Density of networks of systems supported:** Determining the scalability of the CONNECT platform is quite not relevant at this stage given that in its current state CONNECT is still a prototype. However we did some observations in the GMES use case taking into account the number of NSs to discover. First step is to launch all the NSs at the same time. Time to discover all the NSs increases with the number of NSs, and errors of discovery occurs with a couple of hundred of NSs. Though, the CONNECTor generation and the CONNECTor invocation are not impacted. Second step is to launch the NSs gradually. In this case, all the NSs are

discovered whatever their numbers, and CONNECTors generation and invocation are not impacted. Following these observations, strategies to enhance the discovery of a large number of NSs can be applied.

**Distribution of networked systems:** We confirmed in GMES that the networked systems can be distributed in LAN/WAN without problems of CONNECTors-NS communication, considering that the host of the generated CONNECTor is connected to the NS network. In GMES the NS 6 weather service is hosted in a server of TUDO.

# 3 Mobile Collaborative Applications Use Case

## 3.1 Introduction

Current-generation mobile platforms are evolving at a fast pace, with mobile devices embedding an increasing number of innovative features, networking, sensing, or interacting with nearby devices, and new mobile usages being adopted massively in short period of time. We first detail the specific constraints of the mobile domain and their impact on the lifecycle of collaborative applications, and then explain the business needs of Ambientic w.r.t collaborative applications in the mobile domain. We then identify key CONNECT challenges to address in this domain. Finally, we introduce the Ambientic use cases to be fully developed in Section 3.2, and the assessment criteria to be used in Section 3.3 to validate the extensions to the CONNECT architecture that are detailed in Deliverable D1.4.

### 3.1.1 Collaborative applications in the mobile domain

While in the desktop and Internet domains services are loosely bound and easily interchangeable, the mobile domain exhibits very specific constraints and behaviors that go towards integrated and heavily controlled services. Indeed, mobile platform vendors, such as Google with its associated manufacturers with Android, and Apple with iOS, are intensively focusing on the vision of deeply integrating services within the mobile operating systems. This approach raises interoperability issues as older phones are not always updated, for technical or business reasons, and therefore cannot interact with newer phones due to protocol or content format mismatches.

At the same time, mobile devices store and manage much of the user's personal information such as contacts, timetable of personal appointments, location, or payment information that raise privacy-invasion concerns which have led to a number of additional restrictions on mobile platform environments. Most notably, the distribution of mobile applications is heavily controlled, and executable code cannot be generated and deployed at run time. For example, all iOS applications have to pass through a manual screening and approval process before being published on the Apple App Store, which is the only means of application distribution on Apple's platform.

Another major evolution in the design of collaborative applications in the mobile domain is the increasing dependence on Cloud services (or alternatively the availability and use of Cloud services on mobile devices thru native mobile applications). The multiplicity of personal or family connected devices (i.e., tablets, smart TV) is also to be accounted for, which results in the rising demand for exchanging, sharing, and synchronizing data at a global scale between different users and devices.

These major changes in the lifecycle of devices and applications, as well as constraints induced by mobility aspects, such as computational resources, connectivity, or battery life, introduce interoperability barriers between different platforms and also between applications.

### 3.1.2 Ambientic business domain

Ambientic develops solutions to enable collaboration among applications deployed on heterogeneous mobile phone platforms. The first product from Ambientic is a suite of mobile collaborative services, called U-Event, which is aimed at fostering communication among actors at any event (e.g., visitors, exhibitors and panelists at trade show or conference).

Interoperability issues not only arise from the heterogeneity of the mobile devices, but also from the heterogeneity in the Cloud services that mobile users indirectly use from their devices (such as facebook, flick, or google drive), and from the variety of online services that event organizers rely on when setting up their events (such as registrations, badging, matchmaking, or multimedia production).

Apart for the easy integration of Cloud services, Ambientic also aims to offer innovative services that promote interactions among participants and boost the event's profile. Delivering live multimedia services on-site has been identified as such category of services, in particular push-to-talk and video-calling services (small groups), as well the broadcast of conferences' video or audio feeds.

In both cases (mobile and Cloud integration, live multimedia services), we identified significant interoperability problems stemming from the heterogeneity of platforms and services that we need to interconnect with. Ambientic is thus particularly interested in leveraging and further adapting CONNECT solutions to reduce the effort needed to create mobile multi-platform collaborative applications. Specifically, our goals within the project are to assess the applicability of the CONNECT Enablers in the mobile domain, assess the CONNECTion of mobile networked systems towards sustaining mobile collaborative applications, and to support on-the-fly interoperability of streaming protocols.

Ambientic then aims to leverage the improved CONNECT architecture to enable the automatic integration of services into any event IT context, and therefore speed-up the integration process with event organizers. We also aim to quickly deliver compelling collaborative applications that empower users to share content regardless of their terminals or online service providers.

### 3.1.3 CONNECT Challenges for collaborative applications in the mobile domain

CONNECT aims to deliver eternal CONNECTivity to networked systems, primarily through the dynamic synthesis and deployment of CONNECTors that overcome the interoperability gap between these systems. To achieve this outstanding goal, CONNECT has been addressing the following challenges:

1. Modeling and reasoning about peer system functionalities;
2. Modeling and reasoning about CONNECTor behaviors;
3. Runtime synthesis of CONNECTors;
4. Learning CONNECTor behaviors;
5. Dependability assurance;
6. Performant system architecture;
7. Experimenting in the field of wide area, highly heterogeneous systems where today's solutions to interoperability already fall short (e.g., systems of systems).

In the mobile environment, the extensive heterogeneity of mobile networked-systems combined with the fragmented support of legacy protocols and the specific lifecycle of devices and applications, induce an increased complexity for developing multi-platform applications. Indeed, this requires substantial work on integrating with other systems, services and protocols.

This is why we believe that the challenges addressed by CONNECT for achieving eternal interoperability perfectly fit the need of Ambientic, with respect to our sphere of interests, for building Cloud-enabled, cross-platform, and performant Mobile Collaborative Applications. Taking into account the time available, during Year 4, we gave priority to the following CONNECT challenges, displayed in order of importance for the mobile domain:

1. Experimenting in the field of wide area, highly heterogeneous systems;
2. Modeling and reasoning about peer system functionalities;
3. Modeling and reasoning about CONNECTor behaviors;
4. Performant system architecture;
5. Runtime synthesis of CONNECTors.

Given the nature of mobile platforms, we strongly believe that experimentation of interoperability solutions with actual services and business cases on heterogeneous mobile platforms (challenge 1) provides the necessary feedback for enabling more efficient modeling

and reasoning of mobile component functionalities (challenge 2) and required CONNECTor behaviours (challenge 3).

The performance of interoperability solutions (challenge 4) is an equally important concern when dealing with resource-constrained devices. This issue is further accentuated when the mediated systems impose real-time constraints, like, for example, in the case of live multimedia streaming.

As described in Deliverable 1.4, dynamic code deployment on existing mobile platforms is generally prohibited. This constraint introduces further technical and architectural challenges for the run-time synthesis and deployment of CONNECTors on mobile devices (challenge 5).

### 3.1.4 Ambientic use cases

In correlation with Ambientic business needs, we have implemented four use cases, which are representative of the innovations we aim to introduce in the U-Event platform:

- In order to provide, through U-Event, conference's attendees and speakers the capability to use their smartphones to broadcast audio/video streams, we implemented two prototypes that highlight how CONNECT technologies help to achieve multimedia stream interoperability. The former is on mobile video streaming interoperability and highlights Control Protocol as well as Media Container interoperability. The latter, called Push2Talk, implements a "Walkie-talkie" application to be used among a group of people and highlights mobile group communication and interoperability with legacy audio clients.
- Another innovation consists of enabling U-Event to get access to heterogeneous cloud providers to allow event participants to retrieve their document and data from the cloud and share them with other participants. To do so, we implemented a third use-case that demonstrates inter-application mediation and allows a prototype of mobile application to interact with the most popular legacy Cloud services.
- The U-Event application interacts with the Ambientic server, which is responsible to manage users' data and further bridging the U-Event application with upcoming event services. Currently, the bridging process is handmade and time consuming. This is why we envisioned to experiment CONNECT technologies through a real business case involving different event management systems in order to enable a dynamic interoperability including complex interaction protocol stacks with cross-layer dependencies.

### 3.1.5 Assessment criteria

Based on the above challenges and objectives, we define five assessment criteria that allow us to evaluate the potential of CONNECT to deliver added value in the development and usage of mobile collaborative applications:

1. **Interoperability between mobile deployed applications** assesses the ability of CONNECTors to mediate mobile collaborative applications that are either co-located or installed on separate mobile devices.
2. **Interoperability between mobile applications and cloud services** assesses the ability of CONNECTors to enable mobile applications to get access to miscellaneous remote cloud and networked services.
3. **Improvement of the mobile application development process** evaluates the automation impact of the CONNECTor synthesis process on the development of mobile collaborative applications.
4. **Handling of mobile context dynamicity** assesses the ability of collaborative mobile applications to handle changing network conditions inherent to mobile environments.
5. **Mobile application scalability** evaluates mobile communication scalability in term of architecture and usage.

| Criteria/Use Case | Live video streaming | P2T | Cloud Storage | Event Management |
|---|---|---|---|---|
| 1 | ✔ | ✔ | ✔ | |
| 2 | | | ✔ | ✔ |
| 3 | ✔ | ✔ | ✔ | ✔ |
| 4 | ✔ | ✔ | ✔ | |
| 5 | ✔ | ✔ | | |

**Table 3.1**: Use Cases Highlighted Criteria

## 3.2 Mobile collaborative applications use cases

In this section, we detail the design and implementation of the mobile collaborative applications use cases, which serve to assess the aforementioned criteria (see Table 3.1) and demonstrate that CONNECT technologies can be successfully applied in the mobile domain.

### 3.2.1 Mobile video streaming interoperability

Last year, in Deliverable D1.3, we addressed the challenges of enabling Live Multimedia Streaming on heterogeneous mobile devices with reference to the CONNECT architecture. We introduced AmbiStream, a compile-time, multi-platform CONNECTor that can be deployed in fully distributed mobile environments. AmbiStream is based on the iBICOOP middleware technology, a partial and lightweight CONNECT Enabler implementation intended to simplify the development of Collaborative Mobile Applications on heterogeneous devices. iBICOOP was then improved and integrated with the AmbiStream CONNECTor in order to fulfill the goal of interoperable live streaming on current generation mobile platforms. Following the layout described in Figure 3.1, during the third year review demonstration session, we showed that a legacy HLS (HTTP Live Streaming Protocol) Video Player (e.g., an Apple iPad Tablet), can connect and display video streamed by the UAV NS (RTSP Server) using the Real-time Streaming Protocol. The two NSs interoperate via a Mobile CONNECTor deployed on an Android smartphone.

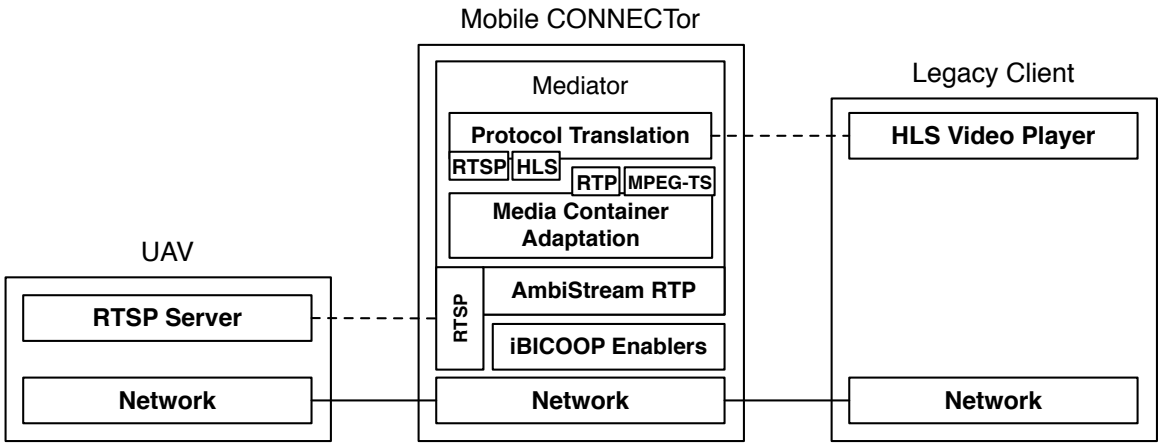


**Figure 3.1***:* AmbiStream Mobile CONNECTor Architecture

For this reporting period, we further extended the AmbiStream mobile demonstrator to enable live streaming interoperability between the NSs that are part of the global GMES scenario. As

described in Figure 3.2, we demonstrate live video streaming interoperability between the UAV video-stream service and the C2 GIS integrated video player. At the same time, the AmbiStream mobile demonstrator can discover, connect, and redistribute the stream of the IP Cameras part of the GMES scenario. This is achieved by relying on the iBICOOP Enablers implementation to overcome the mobility-induced restrictions.

Initially, the AmbiStream Mobile CONNECTor was designed to solve interoperability between live streaming protocols at two levels: *Control Protocol* (represented by the Protocol Translation layer in Figure 3.2) and *Media Container Adaptation*. Each mediation phase is achieved using high-level descriptions (i) of the interaction protocol, and (ii) of the media container format. While mediation done at these two levels is sufficient[2] to achieve interoperability between most live streaming protocols supported by mobile platforms, there exist a number of protocols, which are not agnostic to the image, or audio codec used. In such cases, the video frames (or audio samples) have to be transcoded to the encoding supported by the legacy protocol. This is the case for the C2-GIS integrated video player, which only supports the JPEG image format, while the UAV service provides video frames encoded using the H.264/AVC codec.

Low-level image processing is a resource intensive task, and can only be achieved on current generation mobile devices by using dedicated hardware. Since mobile platforms do not provide the necessary API, or do not support such hardware optimizations, we devised an approach to execute the Image Processing on an external resource-rich *Content Adapter* (see Figure 3.2).

Of course, the *Content Adapter* stack, including the *Image Processing* unit, could be easily deployed on a single resource-rich NS on the local network. But, in order to take mobility into account, we cannot assume that such a system exists or is easily deployable in any network environment. Thus, the *Content Adapter* was designed on top of the *iBICOOP Enablers* layer, which assures transparent communication and discovery for mobile environments. In this way, the adaptation service can be easily deployed as a *Cloud Service* on the Internet, while the mobile CONNECTor can assure interoperability for systems on the local network. Following Figure 3.2, we show that for relaying the multimedia content between the Mobile CONNECTor and the *Content Adapter* service we use the AmbiStream protocol, and RTP media container format.

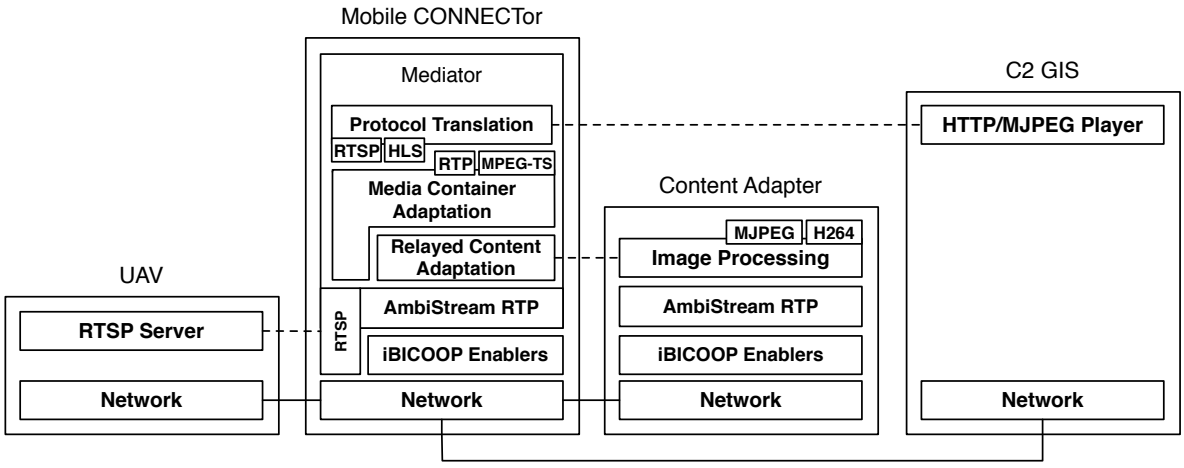


**Figure 3.2***:* Mobile CONNECTor with Relayed Content Adaptation

[2] Emil Andriescu, Roberto Speicys Cardoso, Valérie Issarny: AmbiStream: A Middleware for Multimedia Streaming on Heterogeneous Mobile Devices. Middleware 2011: 249-268

### 3.2.2 Mobile audio streaming interoperability (Push2Talk)

We further experimented live multimedia streaming interoperability, using the AmbiStream CONNECTor, focusing on group communication. For this goal, we have implemented the Push2Talk mobile application. Push2Talk allows one to easily create communication channels where people can meet and discuss in real time, regardless of their phone platform. As seen in Figure 3.3, the graphical interface of the Push2Talk application presents three views: Login, Join/Create channel and the Channel-view. The channel view lists all participants of a particular session. Each participant can push the microphone button to speak.

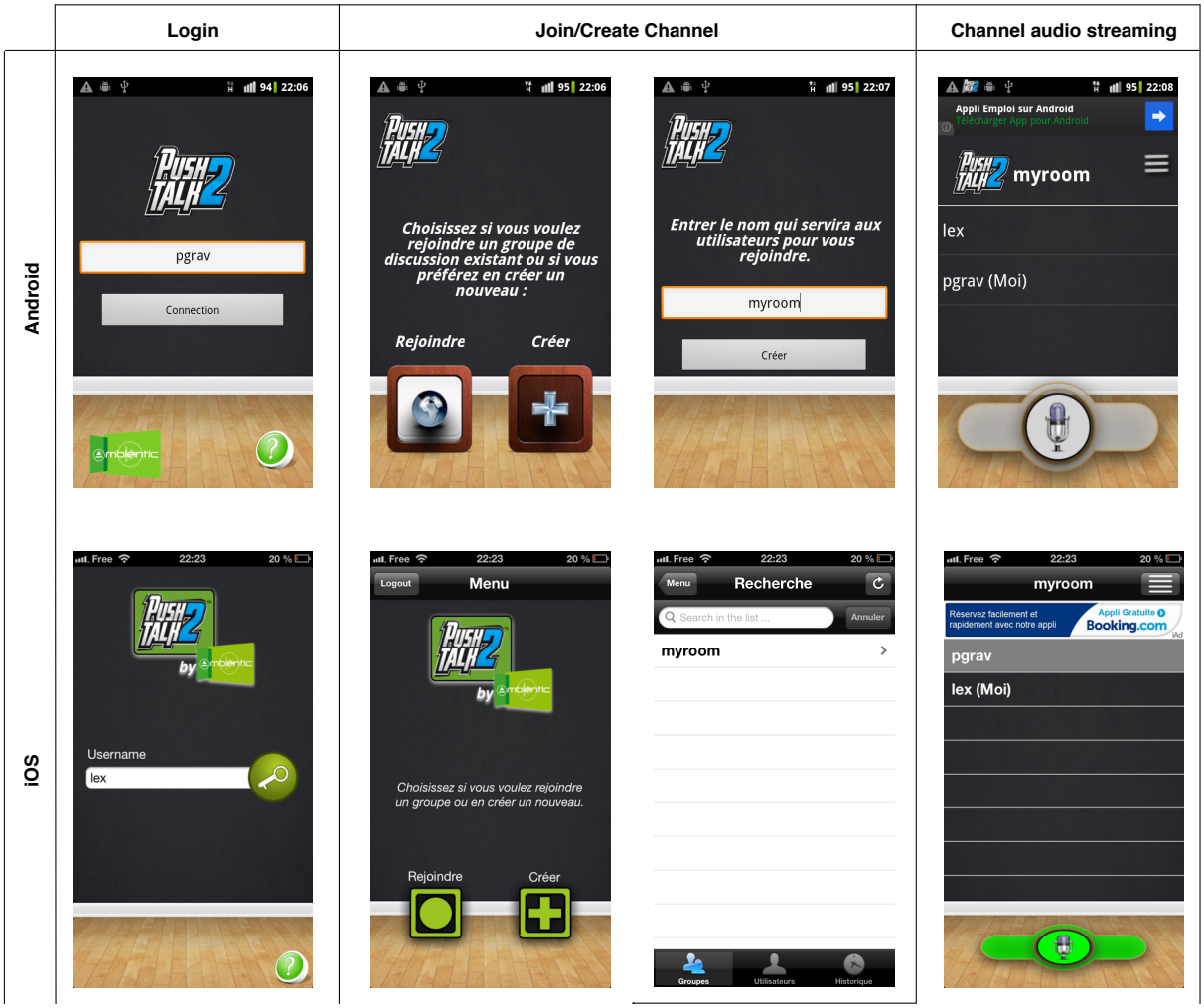


**Figure 3.3***:* Push2Talk application GUI on Android and iOS

In order to evaluate the scalability of the Mobile-Communication Enabler, in a non-simulated environment, we released this prototype as a free application on both Apple and Google marketplaces (i.e., Apple AppStore[3] and Google Play[4]). Since mobile devices are usually not directly addressable over the Internet due to Firewalls and network address translation, Push2Talk relies exclusively on the Relayed Streaming infrastructure, which is part of the lightweight CONNECT Enabler implementation for mobile platforms. Finally, we enable

[3] Push2Talk for iOS: https://itunes.apple.com/en/app/push2talk-connect/id575824793

[4] Push2Talk for Android: https://play.google.com/store/apps/details?id=com.ambientic.push2talk

interoperability between the Push2Talk mobile application and legacy live streaming applications using the AmbiStream CONNECTor. More specifically, we demonstrate interoperability between the Push2Talk application and RTSP-capable streaming clients (e.g., VLC, QuickTime, Android Media Player, etc.).

In the context of CONNECT, the Push2Talk application highlights three main contributions: (i) the mobile cross-platform interoperability (i.e., iOS, Android) assured by the iBICOOP-based CONNECT Enablers; (ii) N2N group communication supporting audio streaming and (iii) interoperability between legacy live streaming protocols (i.e., RTSP) and applications (i.e., VLC) using the AmbiStream CONNECTor. In this use case, the Mobile CONNECTor was deployed packaged along with the Push2Talk application, as described in Figure 3.4.

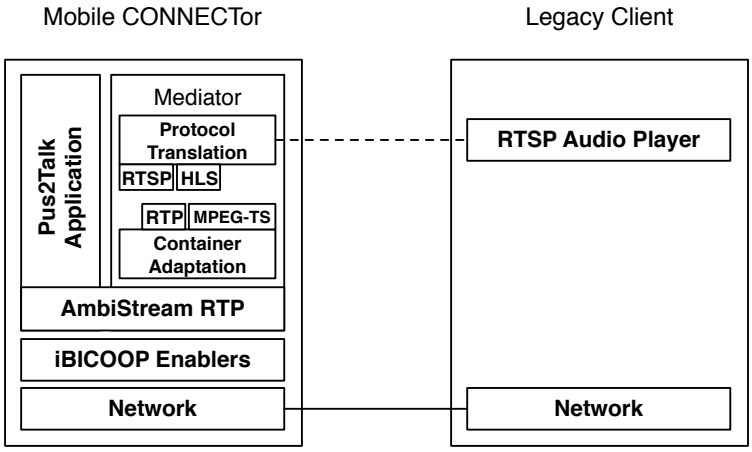


**Figure 3.4***:* The mobile CONNECTor deployed along with Push2Talk application

### 3.2.3 Cloud storage services

Deliverable D1.4 introduces a new way to deploy CONNECTors in the form of mobile applications. The CONNECTor architecture was revised to fit mobile requirements and was also enhanced with a new mobile communication middleware (MiAC) that enables mobile applications to discover and use onboard CONNECTors in order to communicate with co-located applications (i.e. Networked Systems) that were not designed to support such interaction.

We carried out two experiments on the iOS platform. The former involves Instagram legacy applications that use a custom application data format and the latter is an in-house cloud application based on MiAC. It provides more complex behaviors, which demonstrates a more elaborated mediation process.

The first experiment involves the Instagram and Instagram Frame applications both of which have been developed by Instagram. The first application is used to capture, adjust and share pictures. The second application allows pictures to be edited by adding picture frames. Users can also share their framed photos from the Instagram Frame application to the Cloud via the local Instagram application. In order to extend the sharing capability of the Instagram Frame application, we developed a mediator (called *CloudMediator*) which handles the same Instagram data types (i.e., *"com.instagram.exclusivegram"*) and enables the corresponding pictures to be shared with many other cloud applications installed on the mobile device (see Figure 3.4).

When the *CloudMediator* is deployed on the device, the Operating System allows the Instagram application to find the mediator and share images to mediated Cloud services.

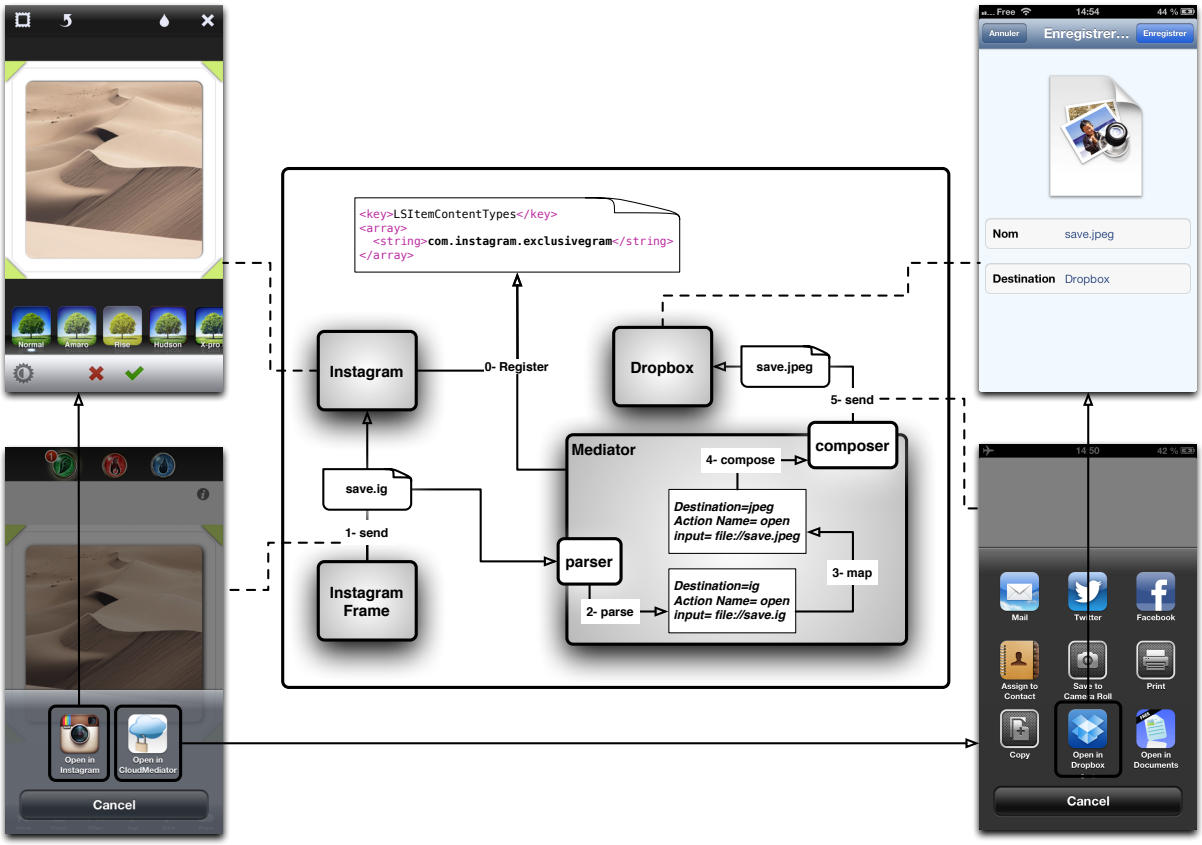


**Figure 3.4: Instagram CONNECTor**

The user can select the *CloudMediator* to share his edited pictures, which drives the *Instagram Frame* application to send files over a proprietary extension “ig” (Figure 3.4 step 1). The CloudMediator receives, parses the file containing the Instagram picture and creates the corresponding abstract action (Figure 3.4 step 2). The Automaton Engine maps the *“open”* Actions and translates the incoming “*ig*" file into a “*jpeg*" file (Figure 3.4 step 4). Finally, the mediator Composer translates the abstract action into a file-call (Figure 3.4 step 4), which triggers the Operating System to display to the user a list of collocated mobile applications which are able to handle the incoming image file via the *CloudMediator*, as for instance Dropbox (Figure 3.4 step 5).

In order to experiment the applicability of the MiAC middleware in the mobile mediation process, we considered the use of Cloud storage services, in particular. Indeed, with the growing usage of mobile applications, many companies provide various Cloud services to help users access their content and synchronize data across their different devices, or interact and share content with other mobile users. The multitude of similar Cloud services, and the heterogeneity of their interfaces make it difficult for mobile applications to leverage these services, either directly or through the mobile applications of the Cloud services vendors. This issue is reinforced when two interacting users intend to use different Cloud storage services.

We designed a demonstrator for the iOS platform that shows how a Cloud-enabled application can seamlessly interact with different Cloud services through mobile mediators. The mobile application referred as *CloudConnect* requires a unique proprietary interface to interact with Cloud storage services. As illustrated in Figure 3.5, this interface defines the following actions:

- ***The Sign-in Action***: enables the application to login and authenticate the user
  - Name: signIn
  - Inputs: email
  - Output: uid (login ok), null (login fail)
- ***Get-user-info Action:*** retrieves user information that is registered in the corresponding cloud
  - Name: userInfo
  - Inputs: uid
  - Output: username, userContact
- ***Get-user-resources Action***: lists all user resources in the related cloud
  - Name: resourceInfo
  - Inputs: uid
  - Output: Array of resourceName

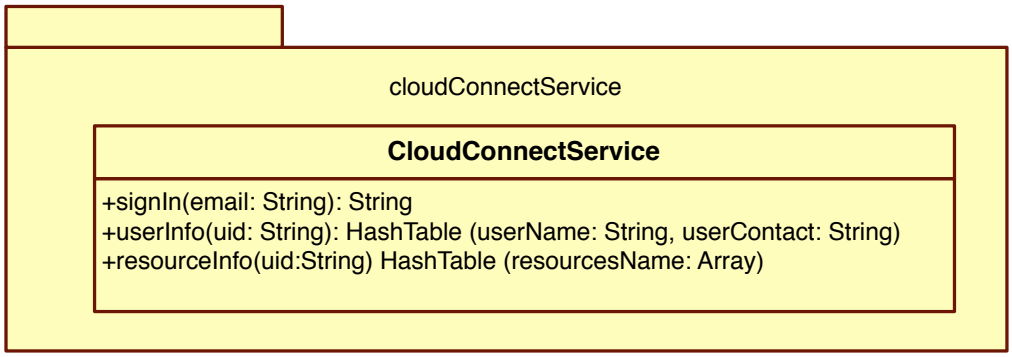


**Figure 3.5: The Required Interface of the CloudConnect Application**

We then designed four instances of the cloud mediator to interconnect the application with major Cloud storage services, namely: Dropbox, Microsoft's Skydrive, GoogleDrive and Flicker.

All these mediators *Register* to handle the *CloudConnect* interface and are able to map each *CloudConnect* action to its corresponding HTTP RESTful action of the mediated Cloud services (Figure 3.6 shows the interfaces for each cloud service).

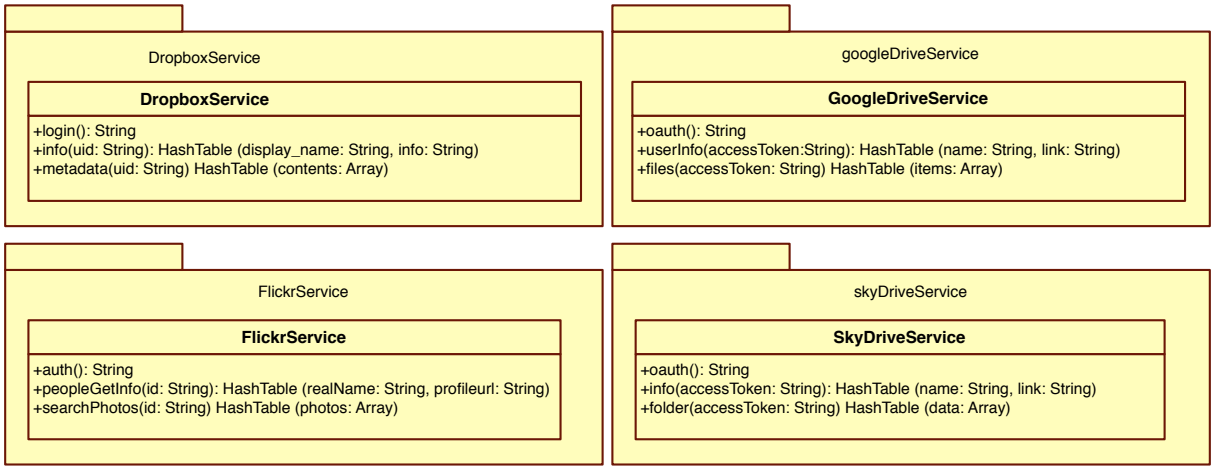


**Figure 3.6: The Provided Interface of the Cloud Service**

Each mediator requires as input an automaton specifying the mapping between the required *CloudConnect* actions and the corresponding cloud service actions.

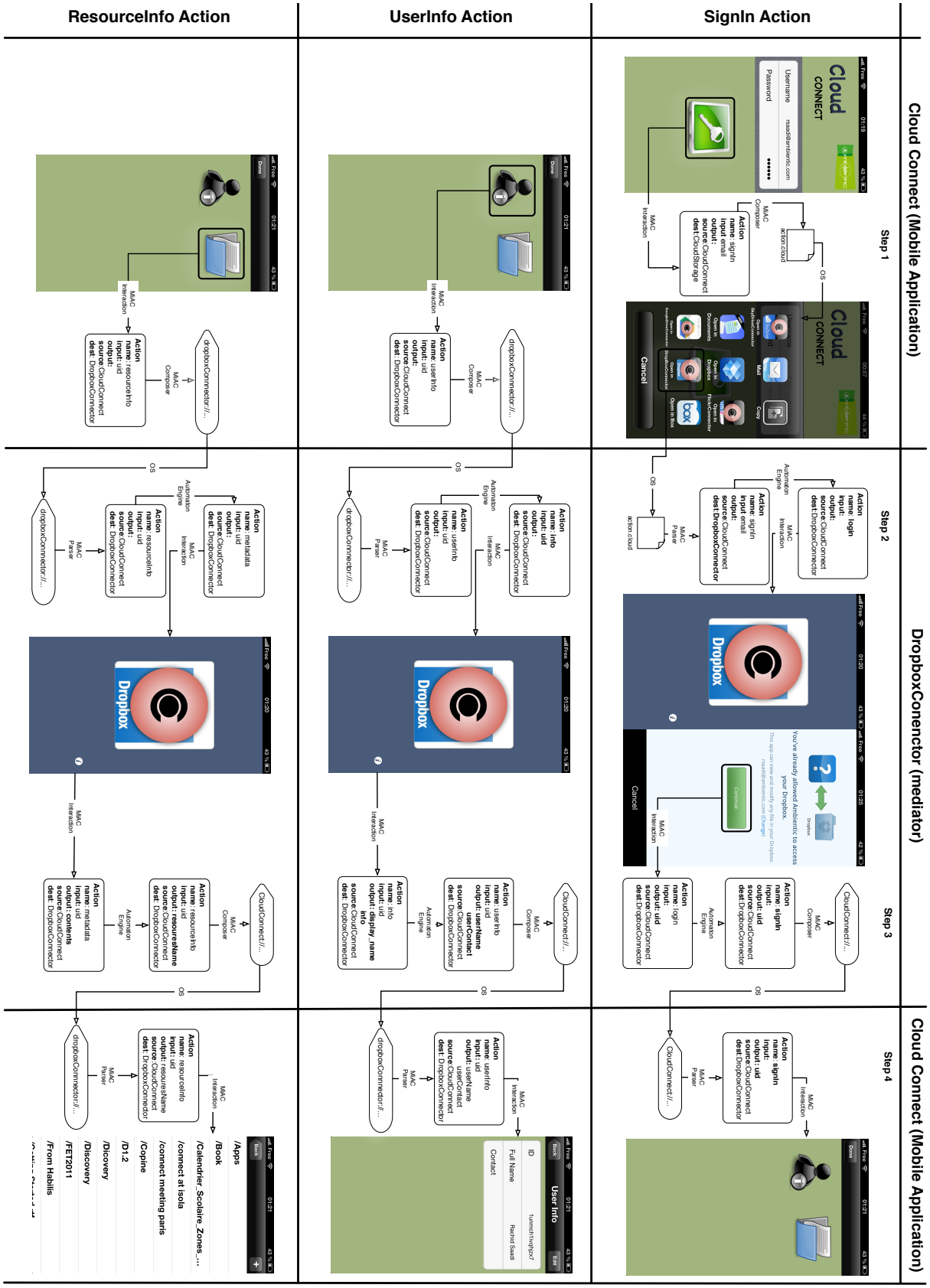


**Figure 3.7: Cloud Mediation Storyboard**

The storyboard in Figure 3.7 shows the step-by-step interaction between the *CloudConnect* application and the *Dropbox* mediator, and highlights how the three aforementioned actions are performed through the mobile mediation middleware. For instance, as the user pushes the authentication button, the application triggers a "*signIn*" action which displays a list of all collocated mediators that were registered for the *CloudConnect* interface (Figure 3.7 step 1). The *Dropbox* mediator is then selected by the user, and further receives the incoming action, which is parsed (by the MiAC Parser) and mapped (by the Automaton Engine) into a *Dropbox* action using the provided mapping (Figure 3.7 step 2). The mediator authenticates users using OAuth [5] process. Then, if the user authorizes the caller application to access his account, the output of the action is filled and mapped back as a *CloudConnect* action before answering the caller (Figure 3.7 step 3). The *CloudConnect* application decodes the action and checks if the initiated "*signIn"* action succeeded or not (Figure 3.7 step 4).

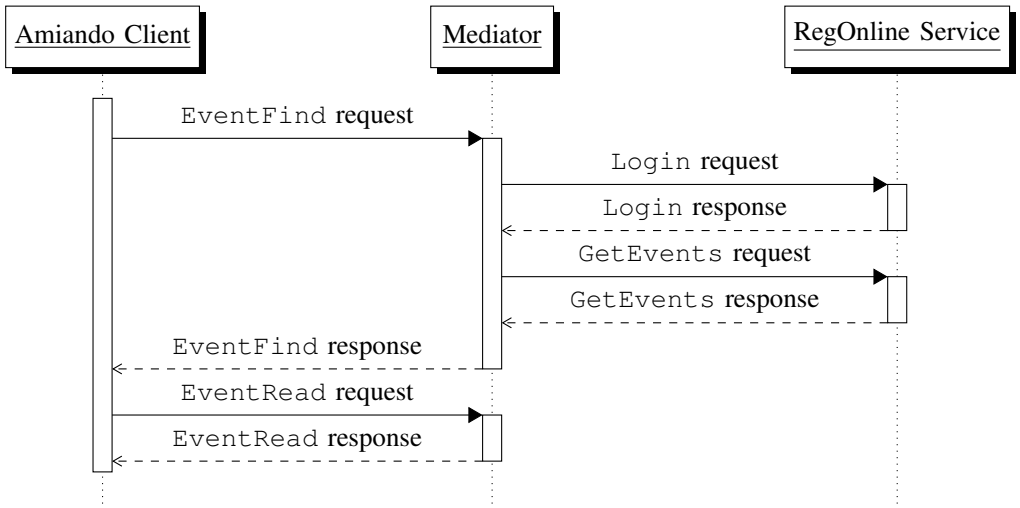


**Figure 3.8***:* Case study interoperability scenario

### 3.2.4 Online event management services

The core business of Ambientic is the Event Market, for which we designed a dedicated mobile application called U-Event. U-Event supports facilitated exchange of data related to an event, coordination of organization tasks, and interaction among all event participants (organizers, exhibitors, booth designers, visitors, the press) via their smartphones.

However, U-event is supported by a web service that enables to store event-data as well as to interoperate with local event services (e.g., the visitor registration server). Currently, for each event we have to manually adapt our web service to interact with different service providers, which is a tedious and time consuming task.

In order to automate the interoperability of different event management systems, we investigated the use of CONNECT technologies. During this experimentation phase we proposed and implemented an improvement to the CONNECT architecture, called FCCL[6], which facilitates cross-layer interoperability by automatically generating parsers and composers for complex protocol stacks. We integrated this framework with the CONNECT Synthesis Enabler,

[5] The OAuth 2.0 Authorization Framework: http://tools.ietf.org/html/rfc6749

[6] FCCL: Framework for Composite Cross-Layer Protocol Interoperability

and were able to synthesize and execute mediators for Regonline[7] and Ambiando[8] event management systems. A detailed presentation of the framework architecture and integration with the Synthesis Enabler is provided in Deliverable D3.4.

In Figure 3.8, we present the interoperability scenario used as a case study. The CONNECTor solves interoperability between the Ambiando client, previously integrated into our application, and the Regonline service that is not supported by U-Event. We show that the mobile application can seamlessly connect to the Regonline service, retrieve and display information about a particular event, via the synthesized CONNECTor. In this case, for privacy reasons, the CONNECTor was deployed on the server side.

## 3.3 Assessment

This section assesses CONNECT for the mobile environment considering the criteria set in Section 3.1.5 and the use cases sketched in the previous section.

### 3.3.1 Interoperability between mobile deployed applications

| **Criteria** | Interoperability between mobile deployed applications. |
|---|---|
| **Initial assessment** | CONNECT architecture lacks support for on-mobile interoperability.<br>Multimedia content has specific interoperability constraints (real-time, fragmentation, multiplexing). |
| **Contribution** | Modeling of live streaming protocol interface.<br>Modeling of mobile inter-app communication.<br>Deployment of CONNECTors on mobile platforms.<br>Mobile CONNECTor architecture (AmbiStream, Mobile Inter-app CONNECTor) |
| **Future work** | Integration of CONNECT Synthesis in the mobile CONNECTor |

Interoperability between mobile networked systems has been experimented for NSs deployed on different mobile devices (live multimedia streaming) or collocated on the same device (Cloud storage using app 2 app communication).

In the live multimedia streaming use cases (video streaming and Push-2-Talk), CONNECTors are deployed and executed on mobile devices, and they support the exchange of video or audio streams between incompatible sources. Specifically, these uses cases, along with GMES, allowed us to experiment with interoperability at different layers:

- **Interaction protocol heterogeneity:** We confirmed that the mobile deployed CONNECTor enables interoperability when interaction protocols differ. For example, the AmbiStream application, deployed on an Android smartphone, is used to translate an RTSP stream coming from the UAV NS to an HTTP Live Streaming stream which is further used to display video in real-time on an Apple iPad.
- **Data format mismatch:** Multimedia content (video frames and audio samples) is sent over the network using various encapsulation methods. In the live video streaming use case, the video frames are adapted from RTP encapsulation to MPEG-TS encapsulation. Besides the media container format, the audio/video samples may also use different compression algorithms (i.e., codecs). AmbiStream enables interoperability between the UAV and C2

[7] Regonline: http://www.regonline.com/

[8] Amiando: http://www.amiando.com/

GIS system, by translating RTSP to MJPEG, which, in turn, requires the transformation of H264 video frames to JPEG images. This final adaptation cannot be done on mobile platforms, so it is achieved using a relayed content adapter deployed on the Internet.

- **Application heterogeneity:** Live streaming applications may have different non-functional requirements. When mediating such interactions, the quality of service might be decreased. In the case of RTSP (real-time stream) to HLS (high-latency stream), mentioned above, the CONNECTor is required to buffer content, thus increasing latency in order to accommodate the incompatible requirements (real-time vs. content buffering).

Interoperability between mobile NSs deployed on the same mobile device has also been assessed in the Cloud storage use case, as different Cloud services may be accessed indirectly thru specific client applications deployed on the mobile device. The experiment on the Apple iOS platform confirmed the feasibility of app2app interoperability, which enables both Cloud service providers and mobile users to better control how third-party applications use these services. We demonstrated how to discover and interact with the Instagram legacy mobile application.

### 3.3.2 Interoperability between mobile applications and cloud services

| | |
|---|---|
| **Criteria** | Interoperability between mobile applications and cloud services. |
| **Initial assessment** | CONNECT architecture lacks support for on-mobile interoperability.<br>Cross-layer dependencies between physical messaging encapsulation layers. |
| **Contribution** | Cross-layer protocol modeling for Cloud service interoperability.<br>Modeling of mobile inter-app communication.<br>Deployment of CONNECTors on mobile platforms.<br>Mobile CONNECTor architecture (Mobile Inter-app CONNECTor).<br>FCCL framework architecture and prototype |
| **Future work** | Integration of CONNECT Synthesis in the mobile CONNECTor. |

Interoperability between mobile NSs and Cloud services has been assessed with the CONNECTor being deployed either on the mobile side (Cloud storage services use case) or on the infrastructure side (Event Management services use case).

As for co-located mobile applications, we have been able to assess the feasibility of the deployment and use, on the mobile, of mediators enabling interoperability between legacy mobile applications with different Cloud storage services.

The event management services use case, allowed us to experiment the deployment and use of CONNECTors supporting mobile applications on the infrastructure side. In this use case, we also integrated and improved the Synthesis Enabler to support complex data types and cross-layer interoperability. Specifically, it enabled us to assess interoperability at different layers to handle the complexity of the interfaces and protocols of the Amiando and Regonline services:

- **Interaction protocol heterogeneity:** Amiando and Regonline services provide incompatible interaction protocols. Based on the behavior specified for each service and their associated interfaces, the Synthesis Enabler was able to identify a correct correspondence between the actions of the two NSs. An example of a valid action correspondence is presented in Section 3.2.4. The synthesized abstract mediator enabled interoperability between the Amiando client, and the Regonline service.

- **Data format mismatch:** Using the FCCL framework, we were able to generate all the required parsers and composers, and their associated SAXSD descriptions. In the context of web services, this level of data adaptation is sufficient to enable valid mapping of values via the abstract mediator.

### 3.3.3 Improvement of the mobile application development process

| | |
|---|---|
| **Criteria** | Improvement of the mobile application development process. |
| **Initial assessment** | Need for rapid integration of Cloud services and in particular Event Management services and storage services.<br>Need for mobile integration of legacy systems. |
| **Contribution** | Modeling of mobile inter-app communication.<br>Deployment of CONNECTors on mobile platforms.<br>Cross-layer protocol modeling for Cloud service interoperability.<br>FCCL framework architecture and prototype.<br>Integration of CONNECT Synthesis for server side support of mobile applications. |
| **Future work** | Integration of CONNECT Synthesis in the mobile CONNECTor architecture. |

From the development standpoint, the use of the different CONNECT technologies was very helpful to speed up the design and the implementation of all use cases.

The AmbiStream prototype (described in D1.3), which is the CONNECT implementation for streaming protocol interoperability in mobile environments, represents the core of the audio/video streaming use cases (i.e., Live Video Streaming, and P2T). Hence, with the help of the synthesis enabler, we were able, without difficulty, to enable support to new streaming protocols (e.g., RTSP, HLS, HTTP/M-JPEG, Ambistream/RTP, etc.) with minor development overhead.

In the cloud storage use case, we based our approach on the mobile CONNECTor architecture, introduced in D1.4, to design a generic CONNECTor skeleton, deployed in the form of a mobile application. By specifying the merged automata that map the CloudConnect application interface with the cloud service interfaces, we were able to synthesize almost integrally all cloud CONNECTor instances (i.e., Dropbox, GoogleDrive, Skydrive, Flickr CONNECTors). Figure 3.9 confirms the implementation effort, since the current CONNECTor generation achieved about three-quarters of the CONNECTor, which is already a satisfactory result. Still, we are now working on integration with the Synthesis Enabler so as to generate the currently handmade automata. We also aim at enhancing the CONNECTor parsers and composers with Starlink or FCCL framework to enable CONNECTors to interact with network interfaces that are supported by legacy middleware such as: REST, SOAP, etc.

The Event management use case is based on the FCCL framework that helps to generate all the required parsers and composers, and their associated SAXSD descriptions to dynamically interact with heterogeneous and complex cloud services. The use of the FCCL framework reduced the development time by (i) enabling the reuse of parser implementations for HTTP and SOAP (ii) facilitating the re-use of the WSDL interface provided by Regonline and (iii) being able to learn the structure of the Amiando JSON-encoded responses, based on provided message samples (initially obtained using a network packet analyzer). However, the mediation process still requires some expert input in the form of high-level models. We believe that part of the input can be further automated by inferring, at least partially, the Message Model by cooperating with discovery mechanisms and packet-inspection software.

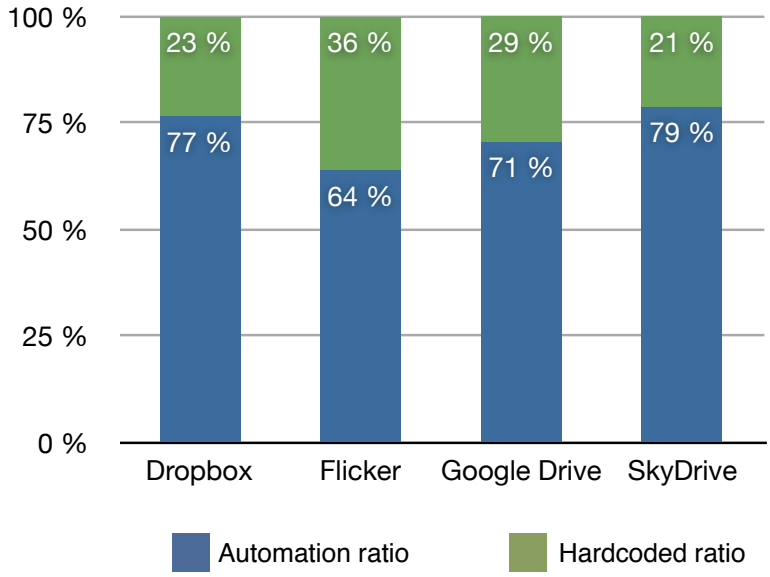


**Figure 3.9**: Static part and dynamic part ratio of the cloud CONNECTor instances.

### 3.3.4 Handling of mobile context dynamicity

| | |
|---|---|
| **Criteria** | Handling of mobile context dynamicity. |
| **Initial assessment** | Supporting horizontal and vertical network handoff on mobile devices.<br>Reachability of the CONNECTors in the mobile environment. |
| **Contribution** | Deployment of CONNECTors on mobile platforms.<br>Mobile CONNECTor architecture (AmbiStream, Mobile Inter-app CONNECTor).<br>Lightweight CONNECT Enablers (Ibicoop Discovery and Communication Enablers) for mobile environments. |

In order to evaluate mobile context dynamicity, we considered three CONNECTor deployment cases, enabling mobile interoperability: iBICOOP-aided deployment, co-located CONNECTor deployment, and shared-context deployment.

First, the Push2Talk application use-case demonstrates the use of the Discovery and Communication Enablers part of the iBICOOP mobile middleware to allow seamless vertical and horizontal network hand-off support for streaming audio data in real-time independently of the network topology and underlying platform. Based on this use-case we demonstrate that users participating in a communication group can seamlessly switch networks and even loose network CONNECTivity for short periods of time, without functional consequences at the application layer.

Second, as we explained in Deliverable 1.4, mobile context dynamicity w.r.t the deployment of CONNECTors can be also achieved by isolation, when the CONNECTors and mediated applications are co-located on a single mobile device. We experimented this type of architecture with the Cloud Storage Services use-case, where CONNECTors were deployed on a single iOS device. We showed that local mobile applications could CONNECT to cloud services via co-located CONNECTors independently of mobile context changes.

Third, we also considered the case where mobile context dynamicity is solved implicitly by assuming that the CONNECTor and the mediated networked systems will share the same context for long periods of time. We demonstrate this case using the AmbiStream application use-case where a legacy video client (an iOS device implementing the HLS streaming protocol) connects to a streaming source (the UAV video service), via a CONNECTor, which is deployed on an Android smartphone. In this case, we assume that the Wi-fi network infrastructure assuring communication between the UAV, the Android-deployed CONNECTor and the iOS device represents a shared network-context.

### 3.3.5 Mobile application scalability

| | |
|---|---|
| **Criteria** | Mobile application scalability. |
| **Initial assessment** | Scaling-up the mobile CONNECTor architecture.<br>Enabling efficient many-to-many communication independently of network topology. |
| **Contribution** | Distributed Mobile CONNECTor architecture.<br>Group communication scalability.<br>Mobile live streaming QoS. |

The AmbiStream CONNECTor relies on the iBICOOP Communication Enabler for assuring scalable live streaming protocol interoperability. This is particularly important in the case of Multimedia Broadcast services. The Push2Talk use case validates this claim by providing real-time N-to-N group communication between hundreds of peers. Further, the Push2talk application was deployed on the Android and Apple application marketplaces in order to validate its performance in real use-case environments.

## 3.4 Summary

In this section, we provided an assessment of the revisited CONNECT architecture that realizes the CONNECT architecture in the mobile applications domain (see Deliverable D1.4). The resulting architecture incorporates a set of contributions that together address the CONNECT challenges relevant to the domain (See Table 3.2).

This assessment as been carried out according to the criteria identified as critical for building collaborative applications in the mobile domain, and the experiments are based on four use cases grounded on Ambientic innovation plans to incorporate in its current business products.

| CONNECT Challenges | Contributions that address the challenge | WP # |
|---|---|---|
| 1 - peer system functionalities | Modeling of live streaming protocol interface | WP1 |
| | Cross-layer protocol modeling for Cloud service interoperability | WP3 |
| | Modeling of mobile inter-app communication | WP1 |
| 2 - CONNECTor behaviors | Deployment of CONNECTors on mobile platforms | WP1 |
| | Mobile CONNECTor architecture (AmbiStream, Mobile Inter-app CONNECTor) | |
| 3 - Runtime synthesis | Integration of CONNECT Synthesis in the mobile CONNECTor architecture | WP1/3 |
| 4 - Performant system architecture | Distributed Mobile CONNECTor architecture | WP1 |
| | FCCL framework architecture | WP3 |
| | Group communication scalability | WP1 |
| | Mobile live streaming QoS | WP1 |
| 5 – Field experiments | Experiments linked to mobile/cloud and Ambientic business | WP6 |

**Table 3.2**: Addressing CONNECT Challenges in the mobile applications domain

# 4 Conclusion

In CONNECT Year 4, the partners in WP6 have provided a suitable platform for assessing the research performed in WP1 – WP5 in a realistic setting. WP6 has collaborated closely with other WPs to produce a GMES use case on top of CONNECT technical and architectural abstractions, while, at the same time, taking into account the feedback received in the previous review.

In the GMES use case, a lot of Networked Systems are involved with heterogeneous communication and data patterns. **This experiment has assessed CONNECT architecture and enablers from a System of System (SoS) perspective, by illustrating and validating the underlying technical approach**. Indeed, we grounded this validation on specific results over:

- The non-intrusiveness of the CONNECT platform over the networked systems;
- Support of Evolutionary development;
- Dealing with systems heterogeneity;
- To deal with geographical distribution of elements in networks of systems.

In addition to this SoS assessment, we introduced a series of Mobile Collaborative interoperability use-cases for highlighting our contributions in WP1 and WP3 on **revisiting the overall CONNECT architecture and prototypes to deal with mobile environments and interaction with associated cloud services**. In this context, we experimented the deployment of CONNECTors on existing mobile platforms (currently, iOS and Android) following the proposed architectural designs for mediating Networked Systems and also mobile inter-application interaction. More specifically, for this second assessment protocol, we looked for assessed results in:

- Interoperability between mobile deployed applications;
- Interoperability between mobile applications and cloud services;
- Improvement of the mobile application development process;
- Handling of mobile context dynamicity;
- Mobile application scalability.

Assessment and validation results for all aforementioned criterions have been precisely characterized in the content of this document.

# 5 Appendix: NS updates from D6.3

## 5.1 Networked Systems

### 5.1.1 NS 1 - UAV

The UAV system is a flying mobile platform, hosting a UAV Camera. It provides a SOAP Web service for its control operations and it offers a RTSP video stream and uses CDP as Discovery Protocol.

#### 5.1.1.1 Interfaces

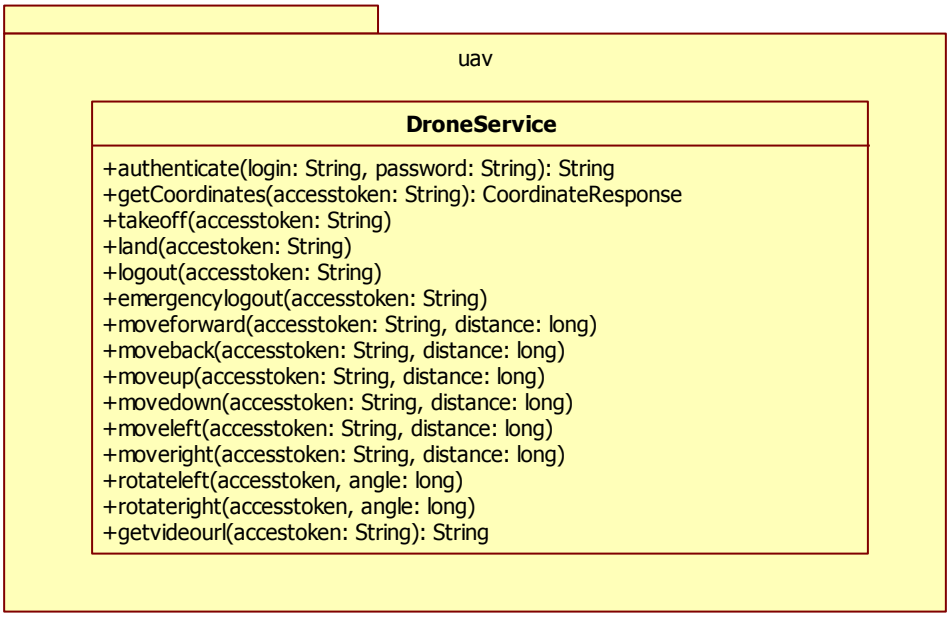


**Figure 1:** UAV Interface

***authenticate****(login: String, password: String): String* – authenticates with a login and password, returns an access token to be used for all other commands

***getCoordinates****(accesstoken: String): CoordinateResponse* – gets current coordinates of UAV, returns a 6D value including roll, pitch, and yaw in addition to X,Y, and Z

***takeoff****(accesstoken: String): Void* – orders the UAV to take off

***land****(accesstoken: String): Void* – orders the UAV to land

***logout****(accesstoken: String): Void* – invalidates the access token

***emergencylogout****(accesstoken: String): Void* – logout from ugv but beforehand, land it.

***moveforward****(accesstoken: String, distance: long): Void* – self explanatory

***moveback****(accesstoken: String, distance: long): Void* – self explanatory

***moveup****(accesstoken: String, distance: long): Void* – self explanatory

***movedown****(accesstoken: String, distance: long): Void* – self explanatory

***moveleft****(accesstoken: String, distance: long): Void* – self explanatory

***moveright****(accesstoken: String, distance: long): Void* – self explanatory

***turnleft****(accesstoken: String, angle: long): Void* – self explanatory

***turnright****(accesstoken: String, angle: long): Void* – self explanatory

***getvideourl****(accesstoken: String): String* – return the video url stream of the embedded camera.

#### 5.1.1.2 Affordance

The UAV's affordance is about providing a vehicle which can move in 3D space. The affordance itself is declared as a subclass of the general "Vehicle" affordance.

```
<?xml version="1.0" encoding="UTF-8"?>
<Affordances name="FlyingMachine"   ontology="connect-gmes-uc.draft5.owl"
middleware="SOAPBinding.xml">
        <Affordance name="FlyingMachineAff" kind="provided">
        <FunctionalConcept>http://www.connect.com/ontology/media#FlyingMachineWithVideo</Functio
nalConcept>
                <Inputs>
                        <Input>http://www.connect.com/ontology/media#Void</Input>
                </Inputs>
                <Outputs>
                        <Output>http://www.connect.com/ontology/media#Void</Output>
                </Outputs>
        </Affordance>
</Affordances>
```

#### 5.1.1.3 Behaviour

The behaviour of the UAV affordance is illustrated in **Figure *2***. The user first needs to authenticate with the service. After authenticating, the user can get the coordinates of the UAV, or order it to move left, right, front, back, up or down, or land. Note that the movements are contingent on first invoking the command for the UAV to take off. Each of the movement operations takes time duration as an argument, which controls how far the UAV will go.

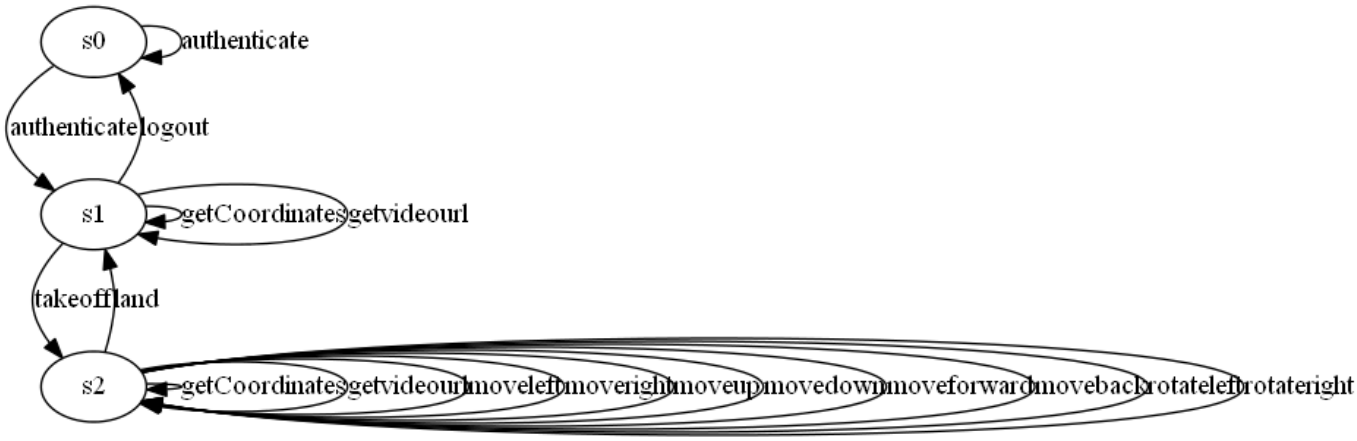


**Figure 2:** Behaviour of UAV

### 5.1.2 NS 2 - UGV

UGV provides access to a video stream issued from a video camera embedded on a ground mobile platform, controlled using remote procedure call (RPC) over HTTP. It uses CDP as Discovery Protocol.

#### 5.1.2.1 Interfaces

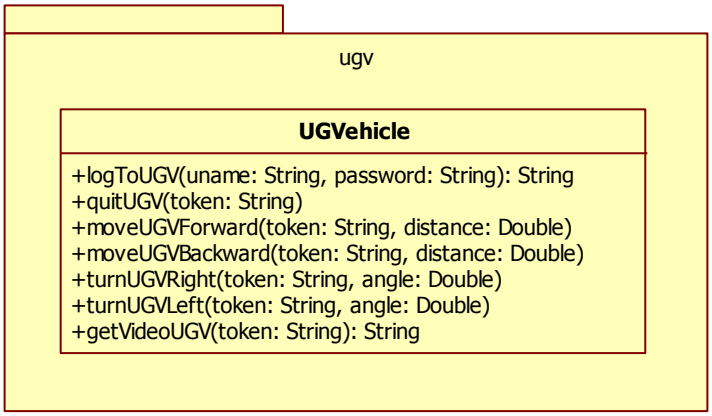


**Figure 3:** UGV interface

***logToUGV****(String uname, String password)* – Log into the UGV with given parameters. Return a valid token if success.

***quitUGV****(String token)* – user kill the session created on UGV.

***moveUGVForward****(String token, Double distance)* - makes the vehicle to move forward of a distance of *distance*. Only authenticated user can use this command.

***moveUGVBackward****(String token, Double distance)* - makes the vehicle to move backward of a distance of *distance*. Only authenticated user can use this command.

***turnUGVRight*** *(String token, Double angle)* - makes the vehicle to turn right of an angle of *angle*. Only authenticated user can use this command.

***turnUGVLeft*** *(String token, Double angle)* - makes the vehicle to turn left of an angle of *angle*. Only authenticated user can use this command.

***getVideoUGV****(String token)* - returns the address of the video MJPEG flux of the camera. Only authenticated user can use this command.

#### 5.1.2.2 Affordance

```
<?xml version="1.0" encoding="UTF-8"?>
<Affordances name="C2Ugv" ontology="connect-gmes-uc.draft5.owl" middleware="SOAPBinding.xml">
        <Affordance name="vehiclevideo" kind="provided">

        <FunctionalConcept>http://www.connect.com/ontology/media#GroundVehicleWithVideo</Functio
nalConcept>
                <Inputs>
                        <Input>http://www.connect.com/ontology/media#Void</Input>
                </Inputs>
                <Outputs>
                        <Output>http://www.connect.com/ontology/media#Void</Output>
                </Outputs>
        </Affordance>
</Affordances>
```

#### 5.1.2.3 Behaviour

The following BPEL explains the sequence of operation permitted on the UGV. Before any action on the UGV, user needs to authenticate with *"logToUGV"*. And then anytime afterwards user can move the UGV with operation *"moveUGV*"* or *"turnUGV*"*. Operation *"getVideoUGV"* can also be called. And finally operation *"quitUGV"* reset the state of the service..

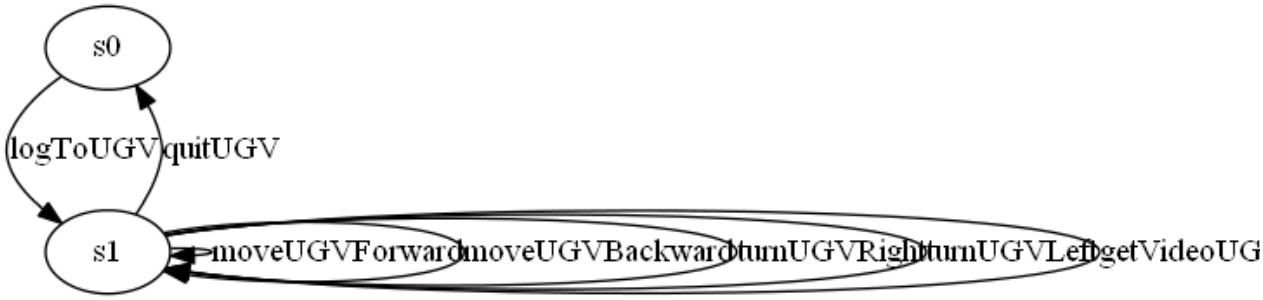


**Figure 4:** UGV Behaviour

### 5.1.3 NS4 - C2 GIS

#### 5.1.3.1 Affordance

C2 GIS needs a lot of different type of services. It displays videos, it controls vehicles and cameras. It needs weather data and position of individuals. The affordances that the C2 declares correspond to request for new network systems. According to the scenario, C2 could be linked with three different types of services connecting to the network, this is reflected in its affordances:: "*WeatherInfo*" for weather data, "*VehicleWithVideo*" for a vehicle hosting a video source, and "*PositioningSource*" for positions feeders.

```
<?xml version="1.0" encoding="UTF-8"?>
<Affordances name="C2Weath" ontology="connect-gmes-uc.draft5.owl" middleware="SOAPBinding.xml">
	<Affordance name="weather" kind="required">

	<FunctionalConcept>http://www.connect.com/ontology/media#WeatherInfo</FunctionalConcept>
		<Inputs>
			<Input>http://www.connect.com/ontology/media#Void</Input>
		</Inputs>
		<Outputs>
			<Output>http://www.connect.com/ontology/media#Void</Output>
		</Outputs>
	</Affordance>
</Affordances>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<Affordances name="C2Ugv" ontology="connect-gmes-uc.draft5.owl" middleware="SOAPBinding.xml">
	<Affordance name="vehiclevideo" kind="required">

	<FunctionalConcept>http://www.connect.com/ontology/media#VehicleWithVideo</FunctionalCon
cept>
		<Inputs>
			<Input>http://www.connect.com/ontology/media#Void</Input>
		</Inputs>
		<Outputs>
			<Output>http://www.connect.com/ontology/media#Void</Output>
		</Outputs>
	</Affordance>
</Affordances>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<Affordances name="C2Pos" ontology="connect-gmes-uc.draft5.owl" middleware="SOAPBinding.xml">
	<Affordance name="position" kind="required">

	<FunctionalConcept>http://www.connect.com/ontology/media#PositioningSource</FunctionalCo
ncept>
		<Inputs>
			<Input>http://www.connect.com/ontology/media#Void</Input>
		</Inputs>
		<Outputs>
			<Output>http://www.connect.com/ontology/media#Void</Output>
		</Outputs>
```

```
        </Affordance>
</Affordances>
```

### 5.1.4 NS 5 - Mobile Weather Station

This system provides local weather information through a shared data space exchange implemented with Lime. It uses CDP as Discovery Protocol.

#### 5.1.4.1 Interfaces

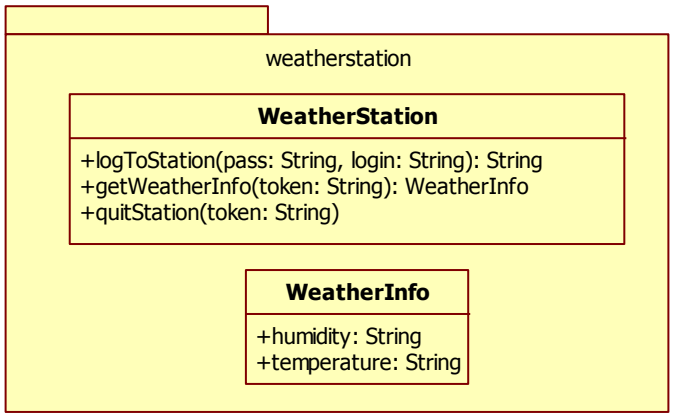


**Figure 5:** Weather Station interface

***logToStation*** *(String pass, String login)* - logs the user into the system using his login/password. It grants the user with a token when login succeeds.

***retrieveTemperatureInformation****(String token)* - returns the current temperature.

***retrieveHumidityInformation****(String token)* - returns the current humidity.

***quitStation*** *(String token)* – destroy the session of the user identified by the given token.

#### 5.1.4.2 Affordance

The weather station has the functional concept *«WeatherInfo*". A client asking for a "*WeatherInfo*" will then be able to connect to this station.

```
<?xml version="1.0" encoding="UTF-8"?>
<Affordances name="WeatherStation">
        <Affordance name="WeatherStationAff" kind="provided">

        <FunctionalConcept>http://www.connect.com/ontology/media#WeatherInfo</FunctionalConcept>
                <Inputs>
                        <Input>http://www.connect.com/ontology/media#Void</Input>
                </Inputs>
                <Outputs>
                        <Output>http://www.connect.com/ontology/media#Void</Output>
                </Outputs>
        </Affordance>
</Affordances>
```

#### 5.1.4.3 Behaviour

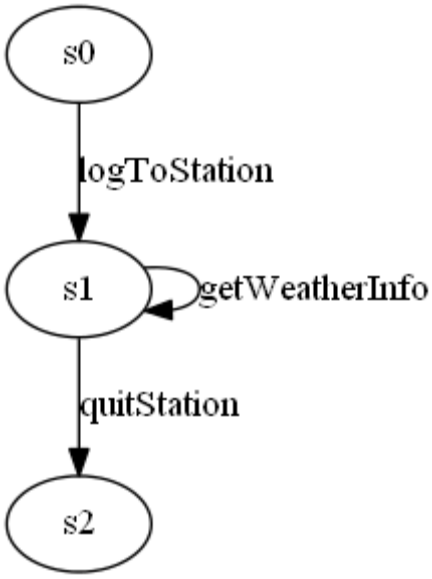


**Figure 6:** Weather station Behaviour

### 5.1.5 NS 6 - Weather Service

This service provides weather report on a given location using a SOAP Web service. It uses CDP as Discovery Protocol.

#### 5.1.5.1 Interfaces

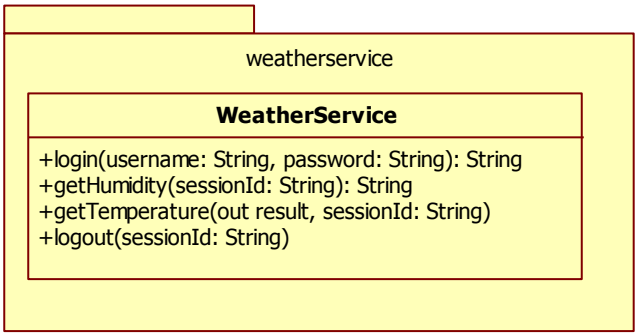


**Figure 7:** Weather Service interface

***login****(String username, String password)* - logs the user into the system using his login/password. It grants the user with a token when login succeeds.

***logout****(String sessionId)* - destroy the session of the user identified by the given token.

***getTemperature****(String sessionId)* – returns the temperature of the location. Need to be authenticated before.

***getHumidity****(String sessionId)* - returns the humidity of the location. Need to be authenticated before.

#### 5.1.5.2 Affordance

The weather service has the functional concept "*WeatherInfo*". A client asking for a "*WeatherInfo*" will then be able to connect to this service.

```
<?xml version="1.0" encoding="UTF-8"?>
<Affordances name="WeatherService">
        <Affordance name="WeatherServiceAff" kind="provided">
```

```
        <FunctionalConcept>http://www.connect.com/ontology/media#WeatherInfo</FunctionalConcept>
                <Inputs>
                        <Input>http://www.connect.com/ontology/media#Void</Input>
                </Inputs>
                <Outputs>
                        <Output>http://www.connect.com/ontology/media#Void</Output>
                </Outputs>
        </Affordance>
</Affordances>
```

#### 5.1.5.3 Behaviour

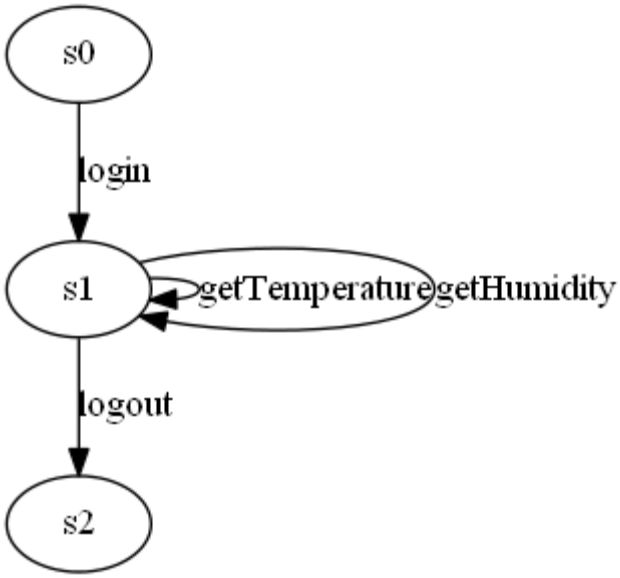


**Figure 8:** Weather service Behaviour

### 5.1.6 NS 7.1 - Positioning System – Country A

This system provides information on location of Country A resources, using a SOAP RPC protocol. It uses CDP as Discovery Protocol.

#### 5.1.6.1 Interfaces

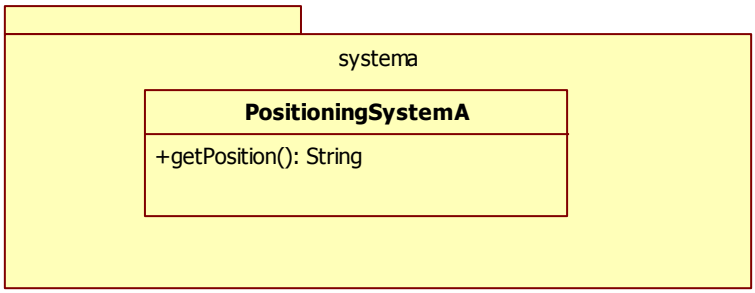


**Figure 9:** Positioning system interface

***getPosition()*** – get the last positions of actors on the fields.

#### 5.1.6.2 Affordance

This service has the functional concept "*PositioningSourceSOAP*" that is a sub class of "*PositioningSource*". A client asking for a "*PositioningSource*" will then be able to connect to this service.

```
<?xml version="1.0" encoding="UTF-8"?>
<Affordances name="PositioningSOAPSystem" ontology="connect-gmes-uc.draft5.owl"
middleware="SOAPBinding.xml">
        <Affordance name="PositioningSOAPSystemAff" kind="provided">

        <FunctionalConcept>http://www.connect.com/ontology/media#PositioningSourceSOAP</Function
alConcept>
                <Inputs>
                        <Input>http://www.connect.com/ontology/media#Void</Input>
                </Inputs>
                <Outputs>
                        <Output>http://www.connect.com/ontology/media#Void</Output>
                </Outputs>
        </Affordance>
</Affordances>
```

#### 5.1.6.3 Behaviour


**Figure 10:** Positioning system A Behaviour

### 5.1.7 NS 7.2 - Positioning System – Country B

This system provides information on Country resource location using an AMQP publish/subscribe. It uses CDP as Discovery Protocol.

#### 5.1.7.1 Interfaces

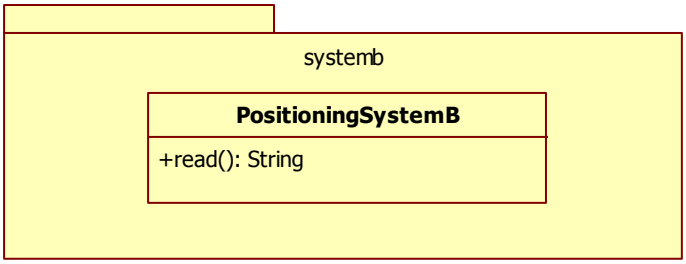


**Figure 11:** Positioning System B interface

***read()*** – get the last positions of actors on the fields.

#### 5.1.7.2 Affordance

This service has the functional concept "*PositioningSourceAMQP*" that is a sub class of "*PositioningSource*". A client asking for a "*PositioningSource*" will then be able to connect to this service.

```
<?xml version="1.0" encoding="UTF-8"?>
<Affordances name="PositioningAMQPSystem" ontology="connect-gmes-uc.draft5.owl"
middleware="AMQPBinding.xml">
        <Affordance name="PositioningAMQPSystemAff" kind="provided">

        <FunctionalConcept>http://www.connect.com/ontology/media#PositioningSourceAMQP</Function
alConcept>
                <Inputs>
                        <Input>http://www.connect.com/ontology/media#Void</Input>
                </Inputs>
                <Outputs>
                        <Output>http://www.connect.com/ontology/media#Void</Output>
```

```
                </Outputs>
        </Affordance>
</Affordances>
```

#### 5.1.7.3 Behaviour

This service is relatively simple. It can respond to any operation "*subscribePositioning*" or "*publishPosition*".

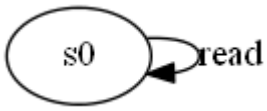


**Figure 12:** Positioning system B Behaviour